PERHITUNGAN KAPASITAS BATERAI DAN ARUS KOMPONEN PADA AR.DRONE *QUADCOPTER* UNTUK ESTIMASI WAKTU DAN JARAK TERBANG

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Yusril Dewantara
NIM: 135150301111093

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PERSETUJUAN

PERHITUNGAN KAPASITAS BATERAI DAN ARUS KOMPONEN PADA AR.DRONE *QUADCOPTER* UNTUK ESTIMASI WAKTU DAN JARAK TERBANG

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Yusril Dewantara
NIM: 135150301111093

Skripsi ini telah diuji dan dinyatakan lulus pada
8 Januari 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Gembong Edhi Setyawan, S.T., M.T	Barlian Henryranu Prasetio, S.T., M.T.
NIK: 201208 761201 1 001	NIK: 201102 821024 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP. 19710518 200312 1 001

IDENTITAS TIM PENGUJI

Penguji 1 / Ketua

1. Penguji 1 / Ketua Majelis

   Mochammad Hannats Hanafi Ichsan, S.ST, M.T

   NIP/NIK : 2016078704231002

2. Penguji 2

   Dahnial Syauqy, S.T., M.T., M.Sc.

   NIP/NIK : 2016078704231002

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata di dalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 8 Januari 2018

Yusril Dewantara

NIM: 135150301111093

DAFTAR RIWAYAT HIDUP

DATA PRIBADI

Nama	: Yusril Dewantara
Tempat, Tgl Lahir	: Kediri, 31 Agustus 1995
Jenis Kelamin	: Laki-Laki
Agama	: Islam
Kewarganegaraan	: Indonesia
Status	: Belum Kawin
Alamat	: Dusun Bancangan Desa Blaru RT.01 RW.13 Kab. Kediri
Telephone	: -
Hp/WA	: 085649035145
Email	: yusrildewantara2@gmail.com

PENDIDIKAN FORMAL :

2001 – 2007	SD Negeri Badas 2
2007 – 2010	MTs Negeri Pare 1 Kediri
2010 – 2013	MAN 3 Kabupaten Kediri

KATA PENGANTAR

Assalamualaikum Wr. Wb

Syukur alhamdulillah penulis ucapkan atas kehadirat Allah Tuhan semesta alam yang mengatur segala sesuatu, yang telah memberikan hidayah, kebaikan, serta memberikan pemikiran yang jernih sehingga penulis mampu menyelesaikan skripsi di bidang komputer ini dengan judul "PERHITUNGAN KAPASITAS BATERAI DAN ARUS KOMPONEN PADA AR.DRONE *QUADCOPTER* UNTUK ESTIMASI WAKTU DAN JARAK TERBANG".

Dalam penyusunan dan penelitian skripsi ini tidak lepas dari bantuan, dukungan, dan saran yang diberikan dari berbagai pihak, maka penulis mengucapkan banyak terima kasih kepada:

1. Bapak Gembong Edhi Setyawan, S.T., M.T., selaku dosen pembimbing satu yang selalu tegas dan bijak membimbing. Selalu membantu dengan ilmunya yang mampu menyelesaikan permasalahan yang penulis hadapi saat sedang mengerjakan skripsi ini. Baik masalah teknis sistem yang dikembangkan ataupun masalah dalam penulisan naskah skripsi.
2. Bapak Barlian Henryranu Prasetio, S.T., M.T., selaku dosen pembimbing dua yang telah memberikan ilmu, saran, serta membantu dalam penyusunan laporan penulis.
3. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D. selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
4. Bapak Heru Nurwarsito, Ir., M.Kom. selaku Wakil Dekan I Bidang Akademik Fakultas Ilmu Komputer Universitas Brawijaya Malang.
5. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D. selaku Ketua Jurusan Teknik Informatika Universitas Brawijaya Malang.
6. Seluruh civitas akademika Informatika Universitas Brawijaya dan teman-teman Teknik Komputer Angkatan 2013 yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Teknik Komputer Universitas Brawijaya dan selama penyelesaian skripsi ini.
7. Bapak Choiril Anam selaku ayah, Ibu Munawaroh selaku ibu dari penulis dari penulis, Arif Dian Kartikasari selaku kakak, Ganes Satria Wardhana sebagai adik penulis, dan Isnaini Almaulida Umami yang selalu mendoakan, selalu menyayangi, dan selalu memberikan dukungan yang amat sangat luar biasa.
8. Ayang Setiyo Putri, Dimas Angger, Faviansyah, Sabitha Wildan, Andyan Bina, Achmad Baichuni, Amroy Casro, Fajar Miftakhul, Latief Nurrohman A, Syarif Hidayatullah, Faisal Lubis, serta seluruh kawan seperjuangan skripsi

quadcopter dan anggota grup skripsi *refresh* yang telah memberikan dukungan, hiburan, dan do'a selama masa pengerjaan skripsi ini.

9. Mochamad Iswandaru, dan Latief Nurrohman Alfansuri yang selalu menemani bercanda, dan membantu finansial penulis.

Penulis memahami bahwa terdapat beberapa kekurangan, tetapi semoga skripsi ini tetap dapat berguna bagi ilmu pengetahuan khususnya dalam dunia teknik komputer. Apabila dimasa depan skripsi ini digunakan sebagai referensi dan acuan penulis akan sangat bangga dan bahagia. Untuk semua itu saya ucapkan terimakasih.

Wassalamualaikum Wr Wb.

Malang, 8 Januari 2018

Penulis

yusrildewantara2@gmail.com

ABSTRAK

Yusril Dewantara, Perhitungan Kapasitas Baterai Dan Arus Komponen Pada Ar.Drone Untuk Estimasi Waktu Dan Jarak Terbang *Quadcopter*

Pembimbing: Gembong Edhi Setyawan, S.T., M.T, Barlian Henryranu Prasetio, S.T., M.T

AR.Drone quadcopter 2.0 adalah salah satu jenis pesawat tanpa awak yang berukuran kecil. *Quadcopter* dapat melakukan perpindahan dengan jarak dan waktu tempuh tertentu yang digunakan dalam *mission flight*, digunakan untuk mencapai lokasi bencana seperti gunung meletus, dan digunakan untuk pemetaan wilayah. Untuk mendukung fungsi tersebut maka dibuatlah sistem memperhitungkan kapasitas baterai dan arus pada komponen yang digunakan untuk dapat mengestimasi waktu dan perpindahan ketika *quadcopter* beroperasi. Dan *quadcopter* beroperasi menggunakan *keyboard laptop* sebagai *controller*. Estimasi waktu terbang dilakukan berdasarkan pada pengitungan arus dari komponen – komponen pada *board quadcopter* dan juga arus pada motor *brushless* yang dikonversi dari *PWM* (*Pulse Width Modulation*) dengan menggunakan persamaan hukum *coloumb*. Untuk estimasi perpindahan dilakukan berdasarkan percobaan pengujian terbang quadcopter yang melakukan perpindahan tertentu dalam waktu satu menit. Hasil dari penelitian ini adalah kesalahan operasi sesuai dengan instruksi dari *keyboard* adalah 0 %, kesalahan ketepatan estimasi waktu adalah 1.14 %, dan kesalah ketepatan estimasi perpindahan adalah sebesar 29.75 %

Kata Kunci : kapasitas baterai, arus listrik, estimasi waktu dan perpindahan*, quadcopter.*

ABSTRACT

Yusril Dewantara, Perhitungan Kapasitas Baterai Dan Arus Komponen Pada Ar.Drone Untuk Estimasi Waktu Dan Jarak Terbang *Quadcopter*

Pembimbing: Gembong Edhi Setyawan, S.T., M.T, Barlian Henryranu Prasetio, S.T, M.T

AR.Drone quadcopter 2.0 is one type of unmanned aircraft that is small. Quadcopter can make moves with certain distance and time traveled used in mission flight, used to reach disaster site such as volcano eruption, and used for mapping area. To support this function, a system that can calculate battery capacities and currents from components used to estimate time and displacement when the quadcopter operates. And quadcopter operates using laptop keyboard as controller. The time flying estimation is based on the current calculation of the components on the quadcopter board and also the current on the brushless motor which is converted from PWM (Pulse Width Modulation) using the coloumb law equation. For estimation of displacement is done based on quadcopter flight test experiments that perform certain displacements within one minute. The result of this research are error of operation according to instruction from keyboard is 0%, the accuracy error of time flying estimation is 1.14%, and the accuracy error of displacement estimation is 29.75%

Key words : battery capacities, electronic currents, estimation, quadcopter.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I PENDAHULUAN

1.1 Latar Belakang

Quadcopter merupakan *UAV* (*Unmanned Aerial Vehicle*) atau pesawat tanpa awak yang mempunyai 4 motor sebagai penggerak. *Quadcopter* memanfaatkan motor *brushless* agar dapat terbang. Motor pada *quadcopter* saling berputar berlawanan arah sehingga menghasilkan gaya tarik dan gaya dorong. Dan *quadcopter* mampu melakukan *take-off* dan *landing* secara vertikal.

Penelitian teknologi dibidang *quadcopter* sedang banyak dikembangkan. Pengembangan banyak menggunakan algoritma sistem cerdas, menggunakan *embedded system*, dan menggunakan pengolahan citra. Seperti penelitian yang dilakukan oleh Hadi (2017) tentang pengendalian *quadcopter* menggunakan *Microsoft Kinnect*, penelitian oleh Jorge (2011) yang mengembangkan *quadcopter* yang terbang dengan mendeteksi dan mengikuti lintasan garis, serta penelitian yang dilakukan oleh Qetkeaw (2011) tentang *self balancing* kamera *quadcopter* dengan tujuan gambar yang diambil *quadcopter* stabil.

Pengguna *quadcopter* sering memanfaatkan *quadcopter* untuk misi terbang (*mission flight*) dengan tujuan mencapai sebuah titik pada jarak dan waktu yang telah ditetapkan. Tetapi pengguna *quadcopter* tidak tahu apakah sisa energi baterai yang digunakan *quadcopter* cukup apabila digunakan untuk terbang sejauh titik tujuan dengan waktu yang telah ditetapkan. Hal tersebut mengakibatkan operasi terbang *quadcopter* menjadi tidak optimal. Maka dari itu dilakukan penelitian untuk menerapkan sebuah sistem yang dapat melakukan *monitoring* energi baterai pada *quadcopter*.

Penelitian tentang penghitungan kapasitas baterai untuk optimasi *quadcopter* pernah dilakukan oleh Turo (2015). Penelitian tersebut menggunakan *BMS (Battery Management System)* dengan memanfaatkan sensor pintar pada baterai. Tujuan dari penelitian yang dilakukan adalah untuk memastikan kondisi awalan kendaraan militer, serta memperkirakan operasi kendaraan militer yang dapat digunakan.

Penelitian lain juga pernah dilakukan oleh Williard, et all (2011) tentang optimasi *mobile robotics*. Parameter yang digunakan untuk melakukan *optimasi mobile robotics* adalah komponen-komponen dari *mobile robotics* yang mengakibatkan proses pengosongan baterai. Penelitian dilakukan dengan mengembangkan *software* dan *hardware* pada sistem memiliki tujuan agar sistem dapat menyajikan data untuk performa *UAV* yang lebih baik.

Dan dalam penelitian ini dilakukan pembuatan sebuah sistem yang dapat melakukan penghitungan kapasitas baterai. Tujuan dari sistem ini adalah dapat mengestimasi waktu dan perpindahan yang dapat dilakukan saat *quadcopter* terbang. Dan hal tersebut dilakukakan agar performa *quadcopter* dapat menjadi lebih optimal.

1.2 Rumusan Masalah

Rumusan masalah – masalah yang dibahas adalah sebagai berikut :

1. Bagaimana cara merancang sistem agar mampu mengestimasi waktu dan jarak?
2. Bagaimana cara mengimplementasikan sistem agar mampu mengestimasi waktu dan jarak ?
3. Seberapa tepat pengendalian *quadcopter* menggunakan *keyboard* laptop ?
4. Seberapa akurat estimasi waktu yang didapat dari pengimplementasian sistem ?
5. Seberapa akurat estimasi perpindahan yang bisa ditempuh oleh *quadcopter* ?

1.3 Tujuan

Tujuan dari sistem yang dibuat adalah :

1. Mengetahui seberapa tepat pengendalian *quadcopter* menggunakan *keyboard*.
2. Mengetahui seberapa akurat estimasi waktu yang didapat dari pengimplementasian sistem.
3. Mengetahui waktu dan perpindahan yang bisa ditempuh oleh *quadcopter* sampai pengisian baterai berikutnya.

1.4 Manfaat

Manfaat yang dapat diperoleh antara lain adalah :

1. Dapat me-monitoring sisa kapasitas baterai, dan indikator dengan menggunakan aplikasi web.
2. Dapat mengoperasikan *quadcopter* secara maksimal, dengan pertimbangan waktu dan perpindahan yang dapat ditempuh.
3. Baterai *quadcopter* menjadi lebih awet.

1.5 Batasan Masalah

Batasan masalah dari sistem yang digunakan adalah :

1. Menggunakan baterai *Lithium Ion Polymer* 3 sel
2. Menggunakan robot *Parrot AR.Drone 2.0*
3. *Monitoring* sistem menggunakan antarmuka *web* dan *web socket*
4. *Quadcopter* dapat dioperasikan diluar ruangan dengan kecepatan angin rendah, dan dapat dioperasikan dalam ruangan yang cukup luas dan sedikit perabotan.
5. Jangkauan terbang *quadcopter* sejauh +/- 20 meter.
6. Menggunakan *input* kecepatan dengan nilai 0.3 hingga 0.7.
7. Estimasi sistem dimulai ketika *quadcopter* hover dengan stabil

1.6 Sistematika Penulisan

Dalam sistematika penulisan ini dibuat untuk dapat memberikan gambaran secara garis besar yang meliputi beberapa bab dalam penelitian yang dilakukan, sistematika penulisan yang akan digunakan antara lain adalah :

BAB 1 PENDAHULUAN

Bab ini membahas mengenai latar belakang masalah, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika penulisan.

BAB 2 LANDASAN KEPUSATAKAAN

Bab ini membahas mengenai landasan teori yang akan menunjang terselesaikannya masalah pada sistem yang dibuat. Dalam bab ini terdapat pembahasan tentang penelitian terdahulu yang akan menjadi acuan, teori mengenai metode yang akan diterapkan serta teori yang berhubungan dengan *hardware* dan *software*

BAB 3 METODOLOGI

Bab ini membahas mengenai metode atau cara-cara yang dilakukan guna menyelesaikan penelitian diantaranya adalah analisis kebutuhan, perancangan, implementasi, dan pengujian secara padat dan jelas. Kemudian tiap tahap metodologi dibahas lebih lanjut pada bab – bab selanjutnya.

BAB 4 ANALISIS KEBUTUHAN

Bab ini membahas tentang apa saja yang dibutuhkan pada sistem yang dibuat, seperti kebutuhan komunikasi, kebutuhan *software* dan *hardaware*, kebutuhan antarmuka, kebutuhan fungsional dan kebutuhan non-fungsional.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Bab perancangan membahas tentang perancangan dari sistem yang akan dibuat, rancangan dimodelkan kedalam bentuk *flowchart* dan perhitungan sesui dengan dasar teori yang ada pada bab landasan kepustakaan. Pada tahap implementasi akan diterapkan dari apa yang telah dirancang sebelumnya.

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini membahas tentang pengujian dari sistem yang telah selesai diimplementasi. Setelah pengujian dilakukan maka selanjutnya data akan dianalisis sehingga diketahui tingkat keberhasilannya.

BAB 7 PENUTUP

bab ini berisi tentang kesimpulan dari sistem yang telah di implementasikan dan di uji. Juga terdapat saran apabila skripsi ini dilanjutkan.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Penelitian yang dilakukan oleh Williard et al (2011) tentang *mobile robotics* menyatakan bahwa fungsi dan kemampuan dari *robot mobile* sering dihambat oleh sumber listrik. Dimana sumber listrik tidak dapat menyediakan energi yang dapat digunakan secara terus – menerus. Oleh karena itu perlu dilakukan optimasi. Salah satu caranya adalah dengan menggunakan penghitungan besaran listrik. Parameter untuk optimasi merupakan arus rata-rata pada saat pengisian dan saat pengosongan. Tujuan dari penelitian tersebut adalah untuk mengetahui siklus periode pengisian sehingga dapat menjadwalkan dan mengantisipasi *over charge* atau *over discharge* pada baterai sehingga kinerja dari *mobile robotics* dapat lebih optimal. Dan hasil yang didapat dari penelitian yang dilakukan adalah sistem dapat mempertimbangkan dan mengatur kehandalan baterai yang disebut dengan sistem *PHM (Prognostic and health management*)

Penelitian oleh Turo (2015) yang melakukan optimasi pada kendaraan militer. Dalam penelitiannya energi baterai dideteksi menggunakan *hardware*, *software* dan menggunakan metode *BMS* (*Battery Management System*). *BMS* memanfaatkan parameter berupa arus listrik yang dikonsumsi oleh kendaraan militer. Arus listrik tersebut diolah menggunakan *Soc* (*State of Charge*) dan *SoH* (*State of Health*). Selain itu juga dilakukan analisis terhadap perilaku kendaraan militer saat *starting* dan saat operasi. Tujuan dari penelitian ini adalah untuk mengetahui status pengisian baterai dan status operasi yang bisa dilakuakan oleh kendaraan militer.

Penelitian yang dilakukan oleh Pengrong (2003) menyatakan bahwa prediksi sisa kapasitas baterai telah didesain pada sebuah sistem dan mempertimbangkan penggunaan daya baterai. Sistem dapat secara efektif mengatur penggunaan daya pada sistem elektronik. *Error* pada sistem cukup tinggi, sehingga pengecekan diminimalisir dengan meggunakan pengukuran arus dan tegangan yang dilakukan secara *realtime*. Hasil yang didapat adalah *error* dari sistem 30% dapat direduksi hingga 20 %.

Dari penelitian yang telah disebutkan, diketahui bahwa penelitian mengenai optimasi mobile robotics dan kendaraan militer dilakukan dengan bermacam-macam cara, tetapi dengan objek penelitian yang sama. Penelitian menunjukkan hasil yang baik dan tingkat kesalahan yang cukup rendah, tetapi tetap terdapat kekurangan.

Penelitian yang menggunakan metode *PHM* (*Prognostic and Health Management*) merupakan metode yang digunakan untuk dapat mendeteksi anomali, mendiagnosis kesalahan, dan dapat digunakan untuk dapat memprediksi waktu penggunaan sistem. Dan kekurangan dari metode PHM adalah sistem yang diimplementasikan akan sangat kompleks dan terdapat kemungkinan terjadi masalah yang disebut dengan *NFF* (*No Fault Found*). Selain itu sistem PHM dianalisis secara manual, sehingga kurang efisien dan memerlukan waktu lebih.

Kekurangan berikutnya adalah sistem tidak dapat secara langsung digunakan pada segala jenis *UAV*. Pada penelitian yang dilakukan unutk *monitoring* kendaraan militer juga memiliki memiliki kekurangan di sisi *interface*, dan analisis sistem dilakukan setelah pengoperasian kendaraan, meskipun data yang diambil *valid* namun akan lebih efektif jika analisis sitem dilakukan secara *realtime*.

Dan pada peneitian dengan tujuan untuk menganaliasa sistem memiliki kekurangan berupa tidak mampunya sistem untuk dapat memprediksi baterai yang digunakan pada sistem yang berbeda, penelitian yang dilakukan hanya bertujuan untuk dapat mengoptimalkan sistem yang dapat memprediksi kapasitas baterai.

Dari kekurangan pada penelitian sebelumnya, dapat diketahui bahwa untuk dapat mengoptimasi *UAV*, dapat dilakukan melalui analisis dan perhitungan besaran listrik yang mempengaruhi kinerja alat. Perhitungan tersebut dilakukan secara *realtime* dengan antarmuka yang cukup menarik dan mudah untuk digunakan. Diharapkan melalui optimasi yang dilakukan, kinerja *quadcopter* akan menjadi lebih optimal. Perhitungan kapasitas dan arus yang dilakukan dalam penelitian pada *quadcopter*, akan menghasilkan sistem yang mampu mengestimasi waktu dan perpindahan saat *quadcopter* sedang terbang.

2.2 Dasar Teori

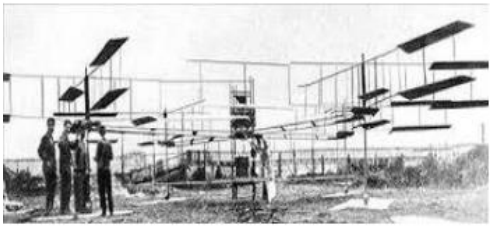
Dalam dasar teori akan dibahas mengenai teori –teori yang berhubungan dengan tujuan sistem dibuat. Dasar teori digunakan sebagai landasan untuk dapat mengimplementasikan sistem sesuai dengan ilmu pengetahuan yang telah dikembangkan.

2.2.1 *Quadcopter*

Menurut hutama (2015) *quadcopter* adalah sebuah sistem yang menggunakan 4 buah motor yang diletakkan pada ujung batang fiber yang ringan yang dibentuk menyilang. Motor yang digunakan biasanya adalah motor *BLDC* (*Brushless Direct Current*) yang mampu mengangkat beban dari perangkat itu sendiri ataupun perangkat lainnya seperti kamera. Pada konfigurasi dengan bentuk plus tiap motor dipasangi dengan baling – baling. Motor depan dan belakang akan berputar searah dengan jarum jam dan motor kiri dan kanan akan berputar berlawanan arah.

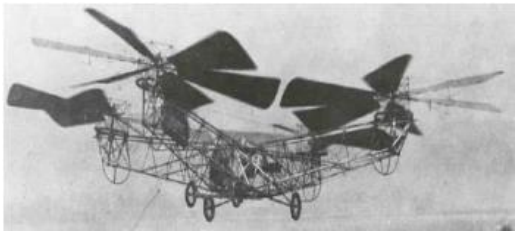
Perkembangan *quadcopter* sedang banyak dikembangkan dalam waktu dekat ini, namun bukan berarti *quadcopter* baru saja diciptakan. Ditunjukkan pada gambar 2.1 bahwa pada tahun 1907 *Breguets* bersaudara merancang sebuah *helicopter* dengan 4 baling - baling. Gambar 2.2 menunjukkan pada tahun 1922 sebuah pesawat dengan model *quadcopter* dirancangan oleh *Georges de Bothezat*. Model yang dirancang memiliki efisiensi lebih dari model sebelumnya. Pada tahun yang sama de *Bothezat* kembali menyempurnakan pesawat dengan model *quadcopter*, *de Bothezat* bekerja sama dengan insinyur bernama *Oemichen*.

Model rancangan mereka ditunjukkan pada gambar 2.3. Model Helicopter dan *quadcopter* dibuat oleh perusahaan penerbangan *Bell Aircraft corporation and the fly vehicles of the Moller company*, perusahaan ini bekerja sama dengan insinyur bernama *Curtiss Wright* pada tahun 1966, atau 3 tahun sejak model rancangan awal yg diciptakan oleh *Curtiss Wright*. Berikut adalah gambaran dari model pesawat atau helicopter yang menggunakan model *quadcopter*.



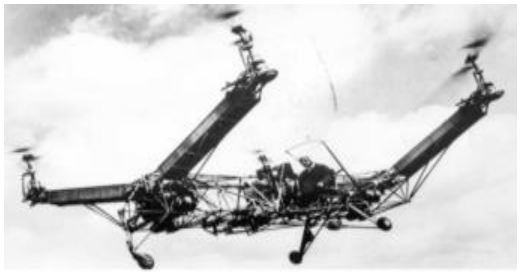
Gambar 2.1 *Breguets* bersaudara

Sumber: Ghazbi (2016)



Gambar 2.2 *Georges de Bothezat*

Sumber: Ghazbi (2016)



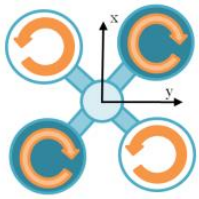
Gambar 2.3 *de Bothezat & Oemichen*

Sumber: Ghazbi (2016)

Gambar 2.4 *Curtiss-Wright*

Sumber: Ghazbi (2016)

Terdapat 2 macam konfigurasi *quadcopter* berdasarkan peletakan koordinat sumbu yaitu konfigurasi *cross* dan *plus*. Konfigurasi tersebut akan ditunjukkan pada gambar dibawah ini :



Gambar 2.5 Konfigurasi *Cross*

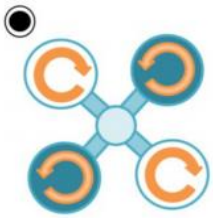
Sumber: Ghazbi (2016)

Gambar 2.6 Konfigurasi Plus

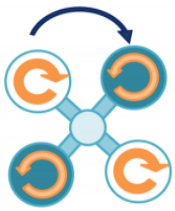
Sumber: Ghazbi (2016)

Cara kerja *quadcopter* adalah dengan 4 motor yang bekerja dengan terdapat putaran yang berlawanan arah. Terdapat 2 buah *motor* yang berfungsi menarik, dan 2 lainnya berfungsi sebagai mendorong. Pada gambar 2.5 ditunjukkan *quadcopter* dengan konfigurasi *cross,* dimana koordinat sumbu X dan Y terdapat di tengah-tengan anatara 2 baling-baling depan dan samping. Juga terdapat *quadcoter* dengan konfigurasi plus yang ditunjukkan pada gambar 2.6 yang merupakan *quadcopter* dengan koordinat sumbu X dansumbu Y berada pada baling-baling depan dan baling-baling samping.

Quadcopter mempunyai 6 *DOF* (*Degree of Freedom*), dimana 4 gerakan translasi dan 2 gerakan rotasi. Operasi terbang *quadcopter* antara lain adalah maju, mundur, naik, turun, ke kanan, ke kiri, *clockwise* dan *counterclockwise*. Operasi maju dan mundur diwakili oleh sumbu X (*pitch*), operasi ke kanan dan kekiri diwakili oleh sumbu Y (*roll*), operasi *clockwise* dan *counterclockwise* diwakili oleh sumbu Z (*yaw*), sedangkan pada naik dan turun diwakili *gaz*.



Gambar 2.7 *Gaz*
Sumber: Ghazbi (2016)



Gambar 2.8 *Clockwise*
Sumber: Ghazbi (2016)

Gambar 2.9 *counterClockwise*
Sumber: Ghazbi (2016)

Gambar 2.10 *Pitch dan Roll*
Sumber: Ghazbi (2016)

Pada *quadcopter,* motor berputar dengan arah yang berlawanan. Dua motor berputar searah jarum jam dan dua motor lainnya berputar berlawanan arah jarum jam. Hal ini ditunjukkan pada gambar 2.7 dalam hal ini operasi tersebut sering disebut dengan *gaz* ketika kecepatan motor berbeda maka akan menjadi operasi *clockwising* dan *counterclockwising* seperti yang ditunjukkan pada gambar 2.8 dan 2.9. Pada gambar 2.10 ditujukkan gerakan *roll* dan *pitch*, *roll* dan *pitch*. *Roll* dan *pitch* dapat terjadi dikarenakan 2 dari 4 motor pada sumbu x berputar dengan kecepatan yang sama.

2.2.2 Elektrokimia

Menurut Crow (1994) elektrokimia berkonsentrasi pada studi tentang pemanfaatan dari perpindahan energi dari satu medium ke medium lainnya. Contoh perpindahan energi dari satu medium ke medium lainnya adalah perpindahan material membran biologi, dan penyimpanan daya baterai.

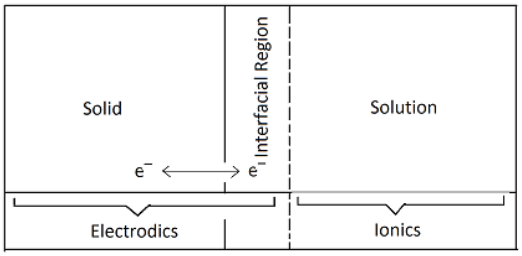
Unit yang sangat penting dalam elektrokimia adalah elektron. Elektron bertugas untuk membawa senyawa ion dalam pertukaran daya antar media. Perpindahan elektron tersebut dipelajari dalam sebuah disiplin ilmu yang dinamakan dengan *thermionics*. Media tempat berpindah elektron adalah senyawa padat yang dimanakan *solid* dan senyawa ionik yang dimakan *solution*.

Dalam *solid* terdapat media konduktor yang berperan untuk melakukan transfer elektron yang dinamakan dengan elektroda. Reaksi yang berada pada

bagian *solid* dimana terdapat elektroda dapat disebut dengan *electrodics*. Sedangkan reaksi kimia yang berlangsung pada bagian *solution* disebut dengan *ionics***.** Reaksi *ionics* dapat terjadi karena mendapat *supply* elektron terlalu banyak sehingga membentuk ion negatif dan ion positif. Melalui penjelasan tersebut dapat disimpulkan bahwa reaksi *electrodics* berkonsentrasi dengan transfer energi untuk memindahkan elektron. Dan Reaksi *ionics* berkonsentrasi pada perilaku ion yang berada pada *Solution*.

Dalam melakukan transfer elektron akan kurang efektif jika hanya menggunakan *solid* dan *solution*. Dapat dikatakan kurang efektif karena jika hanya menggunakan *solid* dan *solution* akan terbentuk luas medan yang tinggi secara ekstrim. Hal itu disebabkan oleh pembebanan beda potensial yang dapat mengurangi jumlah posisi yang dapat ditempati oleh elektron pada *solution* dan *solid*.

Yang akhirnya akan membatasi transfer elektron antar kedua medium tersebut. Untuk itu dibuatlah sebuah pembatas antar kedua medium tersebut yang dinamakan dengan *Interfacial Region*.

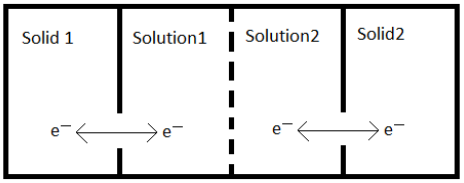


Gambar2.11 Pertukaran Antara Senyawa Elektrodik

Sumber: crow (1994)

Contoh pemanfaatan sel elektrokimia yang paling sering ditemui dalam kehidupan sehari - hari adalah baterai. Terdapat baterai *primer* atau *disposable battery* dan terdapat baterai sekunder yang merupakan *rechargeable battery*. *Disposable battery* dapat dibentuk dari susunan dua buah medium saja yaitu satu *solid* dan satu *solution*.

Sedangkan pada *rechargeable* baterai *solid* dan *solution* disusun sebanyak dua kali dengan susunan solid1 ↔ solution1 || Solid2 ↔ solution2, dan direpresantasikan seperti pada gambar 2.12



Gambar 2.12 Penyusunan Umum Sel Elektrokimia.

Sumber: crow (1994)

Secara spontan *rechargeable battery* menghasilkan tegangan terjadi pada antarmuka antar elektroda/solution dengan pemasangan senyawa kimia yang berbeda. Gangguan pada tegangan eksternal dapat membalik proses pengosongan pada *rechargeable cell*. Persedian energi pada *sector* solution dibawah kondisi elektron tersebut bertukar dan dapat terjadi diantara elektroda dan solution.

2.2.3 Teori Kelistrikan

Pada teori kelistrikan membahas mengenai teori – teori listrik yang berhubungan dengan sistem yang dikembangkan. Pengetahuan yang mendalam tentang teori – teori listrik dalam pengembangan sistem *monitoring* baterai sangat dibutuhkan. Karena acuhan sistem dalam *memonitoring* baterai *quadcopter* adalah melihat besaran – besaran listrik yang digunakan pada *quadcopter*.

2.2.3.1 Muatan Listrik

Muatan listrik adalah suatu satuan pengukuran yang menjukkan jumlah elektron yang mengalir. Muatan listrik dinyatakan dalam satuan *coloumb* dan biasa disingkat (C). Satu *coloumb* setara dengan 6.25 x 10^{28}. Muatan listrik berhubungan dengan arus listrik. Hubungan tersebut dinyatakan dalam persamaan (2.1)

$$I = {}^{Q}/_{t} \qquad (2.1)$$

Dimana :

I = menyatakan Arus (*Ampere*)
Q = Menyatakan muatan (*Coloumb*)
t = Menyatakan Waktu (detik)

2.2.3.2 Tegangan Listrik

Tegangan merupakan ukuran energi potensial yang terletak antara dua titik. Pada listik arus searah, terdapat 2 buah muatan yang berbeda, satu muatan bernilai positif dan muatan lain bernilai negatif. Dari perbedaan dua muatan tersebut sering disebut dengan istilah beda potensial.

Tegangan memiliki satuan *volt*, merupakan satuan yang diberikan untuk menghormati jasa penemu teori tentang tegangan listrik yaitu *Alessandro Giuseppe Antonio Anastasio Volta*. Gaya pada tegangan listrik bisa disebut dengan GGL (Gaya Gerak Listrik) atau dalam bahasa inggris disebut dengan istilah *EMF* (*Electro Motive Force*). Fungsi dari *EMF* adalah memberikann tekanan sehingga arus listrik dapat mengalir dari potensial tinggi ke potensial rendah(dari muatan negatif menuju positif).

2.2.3.3 Arus Listrik

Sirkuit listrik umumnya dibuat dari media yang mampu mengalirkan arus listrik. Elektron akan lebih mudah berpindah pada media konduktif dari pada media isolator. Dengan menggunakan media yang bersifat konduktif maka elektron akan berpindah secara terus-menerus. Perpindahan elektron tersebut disebut dengan arus listrik.

Satuan dari arus listrik adalah *ampere*. Satuan tersebut dipilih untuk menghormati jasa penemu tentang arus listrik yaitu *Andre M. Ampere*. Arus listrik berkaitan dengan jumlah muatan elektron. Seperti yang telah disebutkan dari rumus (2.1) dapat disimpulkan bahwa

1 *Ampere* = 1 *coloumb/second*. (2.2)

Melalui penurunan rumus (2.1) dapat diketahui bahwa

1 *Amper.second* = 1 *coloumb* (2.3)

Dari penurunan rumus (2.2) menjadi rumus (2.3) dapat diketahui bahwa jika kita ingin menuliskan satuan kapasitansi muatan suatu sumber listrik, tidak lagi menggunakan satuan *coloumb* yang bernilai 6.25 x 10^{28} tiap coloumbnya.

2.2.3.4 Hukum Ohm

Setelah mengetahui tentang tegangan dan arus, kemudian pada bab ini membahas mengenai hubungan antara keduanya. Namun untuk mengetahui tentang hukum *ohm* juga perlu mengetahui tentang resistansi atau umunya disebut dengan hambatan.

Menurut Muttaqin (2012) resistansi adalah kemampuan sebuah elemen rangkaian untuk menahan arus listrik. *Resistor* dapat menghambat perpindahan elektron, sehingga dapat mengurangi nilai dari tegangan dan arus sesuai dengan konfugurasi seri atau parallel. Pada rangkaian listrik parallel, tegangan akan bernilai tetap namun arus akan berubah, sedangkan pada rangkaian listrik seri, tegangan akan berubah nilainya dan arus akan bernilai tetap. Jadi kesimpulannya hukum ohm adalah hukum yang menyatakan bahwa akibat dari beda potensial dari dua titik yang terdapat jarak sehingga arus dapat mengalir secara terus-menerus pada suatu penghantar yang memiliki sifat melawan gerakan perpindahan elektron dari potensial tinggi ke potensial rendah.

Tabel 2.1 besaran Hukum Ohm

Besaran	Satuan	Simbol
Tegangan	*Volt*	*E* dan *V*
Arus	*Ampere*	*I*
Hambatan	Ω	*R*

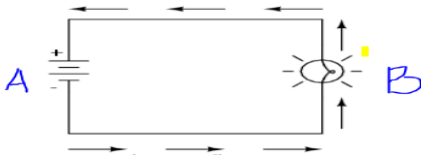
Sumber: Khupaldt (2006)

Tabel besaran hukum *ohm* memuat informasi mengenai satuan serta simbol yang digunakan. Dari tabel tersebut dapat dibuat kedalam bentuk matematis seperti pada persamaan (2.4), (2.5), dan (2.6).

$$V = I \times R \quad (2.4)$$

$$I = {}^{V}/_{R} \quad (2.5)$$

$$R = {}^{V}/_{I} \quad (2.6)$$



Gambar 2.13 Cara Kerja Hukum ohm

Sumber: Khupaldt (2006)

Melalui ilustrasi dari gambar 2.13 dapat diketahui terdapat tegangan listrik yang ditunjukkan oleh huruf A. Hambatan berupa lampu yang diwakili oleh huruf B, dan arah arus yang diwakili oleh tanda panah. Proses lampu dapat menyala di sebabkan oleh gaya gerak listrik atau *electromotive force* yang mengakibatkan arus mengalir melewati penghantar dan beban dari polaritas negatif menuju positif.

2.2.3.5 Daya dan Energi

Dalam analisis sistem yang menggunakan listrik tidaklah cukup jika hanya menggunakan tegangan dan arus, meskipun keduanya merupakan dua satuan yang sangat penting pada disiplin ilmu kelistrikan. Daya dan Energi merupakan satuan lain yang digunakan untuk dapat membantu analisis sistem listrik. Daya adalah energi yang digunakan tiap satuan waktu, dapat dinyatakan kedalam persamaan (2.7) :

$$P = \frac{dW}{dt} \tag{2.7}$$

Dimana :

P = Daya (*Watt*)
W = Energi (*Joule*)
t = Waktu (*Second*)

Daya listrik memiliki keterkaitan dengan tegangan dan arus, pada persamaan (2.8) berikut ini menyatakan hubungan antara daya dengan arus dan tegangan.

$$P = V \times I \tag{2.8}$$

Kemampuan yang digunakan untuk melakukan usaha atau daya bisa disebut dengan Energi. Eneri adalah integral terhadap waktu yang berasal dari satuan daya listrik. Secara matematis daya dirumuskan seperti pada persamaan (2.9) :

$$W = \int P \, dt \tag{2.9}$$

Dimana :

W = energi (*joule*)
P = daya (*Watt*)
t = waktu (detik)

Energi juga memiliki keterkaitan dengan arus dan tegangan. Seperti yang telah dinyatakan pada rumus (2.5) dan rumus (2.6), maka hubungan energi dengan arus dan tegangan dapat dijadikan bentuk matematis seperti pada persamaan (2.10) :

$$W = \int V.I\, dt \tag{2.10}$$

2.2.3.6 Rangkaian Seri dan Parallel

Terdapat dua jenis konfigurasi dalam rangkaian listrik, yakni rangkaian seri dan parallel. Kedua rangkaian ini memiliki *output* tegangan dan arus yang dapat bernilai beda. Pada rangkaian *parallel* tegangan akan bernilai sama dengan sumber tegangan, namun nilai arus akan berbeda. Pada rangkaian seri nilai dari arus listrik akan sama dan tegangan akan bernilai beda. Jika sumber listrik dirangkai seri maka akan menghasilkan tegangan dan daya yang besar, namun jika salah satu jalur diputus maka semua komponen yang berada pada rangkaian hubung singkat tidak akan berfungsi. Rangkaian *parallel* memiliki keuntungan jika salah satu jalur yang terhubung pada suatu komponen maka komponen lain tidak akan terpengaruh. Perbedaan lain adalah rangkaian parallel memiliki simpul sedangkan rangkaian seri tidak. Namun Keduanya sering dikombinasikan. Contoh komponen semikonduktor yang sering dirangkai seri dan parallel adalah resistor, karean setiap komponen elektronik memiliki resistansi.

Pada sistem yang kompleks rangkaian ini sering disederhanakan agar lebih mudah saat menganalisis. Persamaan matematis rangkaian *parallel* dinyatakan pada persamaan(2.11), sedangkan persamaan matematis rangkaian seri ditunjukkan pada persamaan (2.12).

$$Rtotal = R_1 + R_2 + R_3 \ldots . + R_n \tag{2.11}$$

$$\frac{1}{R} total = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \cdots + \frac{1}{R_n} \tag{2.12}$$

Setelah mengetahui cara untuk menganalisis rangkaian listrik dari besar hambatan. Tentunya untuk analisis arus pun menggunakan cara yang mirip. Teori tentang rangkaian listrik menyatakan bahwa rangkaian listrik *parallel* akan menghasilkan arus yang nilainya berbeda dan nilai tegangan yang sama. Sedangkan pada rangkaian seri, arus akan menghasilkan nilai yang sama dan menghasilkan tegangan yang bernilai beda. Pada analisis rangkaian dengan menggunakan resistor, rangkaian seri dimodelkan secara matematis seperti pada persamaan (2.11), namun untuk menganalisis arus yang dimodelkan secara seri rangkaian seri digunakan persamaan matematis (2.13), sedangakan analisis arus pada rangkaian parallel dimodelkan kedalam bentuk persamaan matematis (2.14)

$$I_{total} = I_1 + I_2 + I_3 \ldots + I_n \tag{2.13}$$

$$\frac{1}{I\ total} = \frac{1}{I_1} + \frac{1}{I_2} + \frac{1}{I_3} + \cdots + \frac{1}{I_n} \tag{2.14}$$

2.2.3.7 Hukum Kirchoff

Hukum kirchoff merupakan pengembangan analisis arus dan tegangan listrik. Hukum ini ditemukan oleh ilumuan ahli fisika Jerman yang bernama *Gustav Robert Kirchoff*. Terdapat dua macam hukum kirchof yaitu *KVL* (*Kirchoff Voltage Law*) dan *KCL* (*Kirchoff Current Law*).

Hukum tegangan kirchoff menyatakan bahwa jumlah aljabar dari nilai tegangan pada arah tertentu yang bernilai 0 dan terjadi pada rangkaian hubung singkat. Hukum *kirchoff* dapat dimodelkan kedalam bentuk matematis, seperti pada persamaan (2.15)

$$V_1 + V_2 + V_3 \qquad \sum_{n=1}^{N} Vn = 0 \tag{2.15}$$

Penting untuk diketahui ketika menganalisis dengan hukum tegangan *kirchoff* adalah menentukan polaritas tegangannya. Nilai positif didapat ketika arah dari analisis searah dengan arah tegangan. Dan nilai negatif didapat ketika arah dari analisis berlawanan arah dengan arah tegangan.

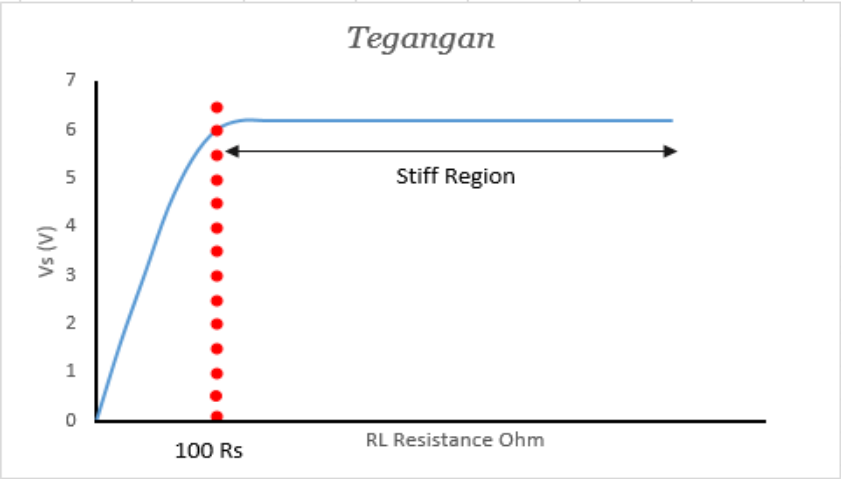
Hukum arus *kirchoff* menyatakan bahwa jumlah aljabar dari arus listrik akan berilai 0 ketika arus keluar atau arus masuk. Hukum arus *kirchoff* dapat dinyatakan dalam bentuk matematis seperti pada persamaan (2.16).

$$I_1 + I_2 + I_3 \qquad \sum_{n=1}^{N} In = 0 \tag{2.16}$$

Sama seperti hukum tegangan *kirchoff* yang harus memperhatikan arah tegangan dan arah analisis. Pada analisis hukum arus *kirchoff* arus yang keluar bernilai negatif sedangkan arus yang mengarah kedalam bernilai positif.

2.2.3.8 Sumber Listrik DC (Direct Current)

Sebuah sumber listrik yang ideal adalah sumber listrik yang mampu memproduksi tegangan beban yang konstan. Tegangan beban akan bernilai konstan karena resistansi beban yang berbeda – beda. Contoh dari sumber listrik ideal adalah baterai yang memiliki resistansi *internal* bernilai 0.



Gambar 2.14 *Stiff Voltage*
Sumber: Malvino (2014)

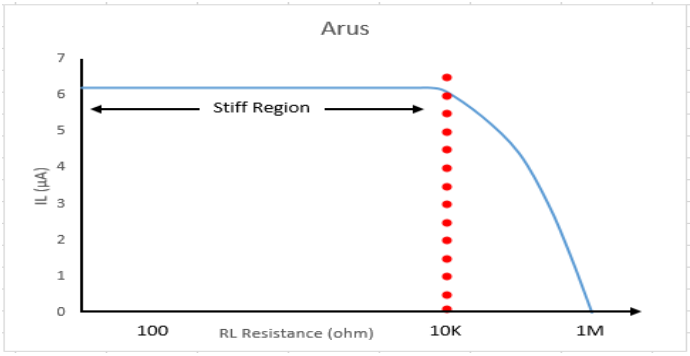
Malvino (2014) menyatakan bahwa suatu Tegangan beban dari sumber listrik akan berada pada kondisi stiff ketika resistansi pada beban 100 kali lipat lebih besar daripada resistansi sumber (*Internal Resistance*). Hal tersebut dapat dinyatakan kedalam bentuk matematis pada persamaan (2.17)

$$Stiff\ Voltage = R_s < 100\ R_L \tag{2.17}$$

Dimana :

R_s = *Resistance in Source*/*Interal Resistance* (Ω)
R_L = *Resistance in Load* (Ω)

Berbeda dengan tegangan, resistansi beban yang berbeda akan menghasilkan arus beban yang konstan. Gambar 2.15 akan menujukkan penyebab perubahan dari arus stiff menuju arus yang berubah ubah dikarenakan *internal resistance*.



Gambar 2.15 Current Stiff
Sumber: Malvino (2014)

Gambar 2.15 merupakan grafik yang menujukkan bahwa ketika resistansi beban suatu sumber listrik bernilai antara 0 hingga 10 KΩ maka nilai arus pada beban akan bernilai konstan, dan apabila nilai dari resistansi beban melibiihi10 KΩ maka nilai dari arus pada beban akan berubah atau turun. Hal ini dapat dimodelkan kedalam bentuk matematis yang ditunjukkan pada persamaan (2.18).

$$Stiff\ Current = R_S > 100R_L \tag{2.18}$$

Dimana :

Rs = Resistansi sumber/ Resistansi Internal (Ω)
R_L = Resistansi beban (Ω)

Baterai dapat membuat elektron mengalir pada sebuah rangkaian listrik melalui pertukaran reaksi kimia dan ionik. Karena baterai merupakan sumber listrik *portable* sehingga suatu saat senyawa kimia yang menghasilkan perpindahan elektron-pun juga akan habis. Hukum *coloumb* digunakan sebagai

satuan kapasitas baterai. Tetapi karena terlalu panjang maka disingkat menggunakan rumus yang telah dituliskan pada persamaan (2.2) dan (2.3), sehingga

$$I = \frac{Capacity\ Rate}{t} \tag{2.19}$$

$$t = \frac{Capacity\ Rate}{I} \tag{2.20}$$

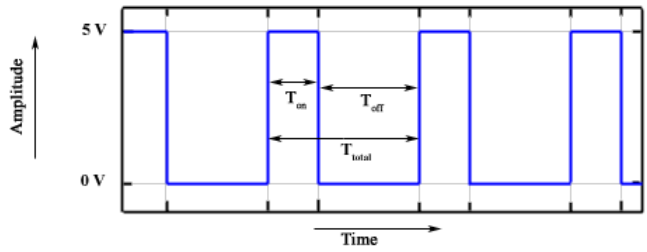
Dengan :

I = menyatakan Arus listrik (*Ampere*)
Capacity Rate = menyatakan kapasitas baterai (*mAh*)
t = menyatakan waktu (*second*)

Sebuah baterai yang mengalami pengosongan tidak hanya disebabkan karena energi yang habis, namun juga dipengaruhi oleh *internal resistance*. Selain dapat mempengaruhi pengosongan kapasitas baterai, resitansi juga mempu menyebabkan tidak benarnya nilai pembacaan dari sisa kapasitas baterai. Dan penyebab paling umum naiknnya resistansi internal adalah temperatur. Suhu yang panas disebabkan oleh energi yang tidak berpindah secara sempurna, sehingga dapat mengakibatkan rusaknya sel baterai, yang dapat menyebabkan baterai menjadi bengkak dan umur baterai yang berkurang.

2.2.4 *Pulse Width Modulation* (*PWM*)

PWM adalah sebuah metode yang digunakan untuk dapat memodulasi lebar pulsa dalam suatu periode, dengan tujuan untuk mendapatkan tegangan rata – rata yang berbeda – beda. Dengan menggunakan PWM digital maka nilai dari PWM didapat dari resolusi PWM itu sendiri. Umumnya PWM digital memiliki nilai maksimal 255, nilai 255 didapat dari 2^n, n disini menyatakan nilai bit pwm, yaitu 8 bit. 255 menyatakan *duty cycle* yang bernilai anatar 0 – 100 %.



Gambar 2.16 *Square Signal PWM*
Sumber: Tutorial Points Pvt. Ltd. (2016)

Square signal pada gambar 2.16 menyatakan *duty cycle*, dimana harga Y menyatakan amplitudo yang merupakan nilai tegangan, dan harga X menyatakan waktu yang memiliki satuan detik. Untuk bisa mendapatkan tegangan keluaran, digunakan persamaan matematis (2.17), (2.18), dan (2.19)

$$Time_{total} = T_{on} + T_{off} \tag{2.21}$$

$$Duty\ Cycle = \frac{Ton}{Ton+Toff} \quad (2.22)$$

$$V_{out} = Duty\ cycle\ \times V_{in} \quad (2.23)$$

Dimana :

T_{On}	= Waktu *square signal* bernilai 1, atau dalam mode *HIGH*
T_{Off}	= Waktu *square signal* bernilai 0, atau dalam mode *LOW*
Duty Cycle	= Resolusi *PWM* (%)
V Out	= Nilai tegangan keluaran dari konversi *PWM* (*volt*)
V In	= Nilai tegangan masukan (*volt*)

Dari persamaan (2.21), (2.22), dan (2.23), nilai *duty cycle* dapat disederhanakan dengan persamaan (2.24). 255 merupakan nilai maksimal dari *pwm 8 bit*, sehingga dapat diketahui bahwa nilai *duty cycle* dari 255 adalah 100%.

$$Duty\ Cylce = \frac{nPwm\ x\ 100}{255} \quad (2.24)$$

Sedangkan untuk bisa mendapatkan nilai tegangan adalah dengan menggunakan persamaan (2.25)

$$Voltage = \frac{Duty\ Cycle}{100}\ x\ Vinput \quad (2.25)$$

2.2.5 Kinematika Dalam Satu Dimensi

Kinematika adalah penjelasan menganai bagaimana sebuah objek dapat bergerak dan berpindah. Salah satu hal yang dibahas dalam kinematika adalah perpindahan posisi yang disebabkan perubahan kecepatan dan perubahan waktu. Perpindahan adalah perubahan posisi sebuah benda dari titik acuan yang merupakan besaran yang mempunyai besar dan mempunyai arah. Waktu merupakan besaran yang menyatakan seberapa lama benda berpindah dari titik acuannya. Dan kecepatan membahas mengenai seberapa cepat sebuah benda dan membahas arah gerakannnya. Bentuk matematis dari perpindahan, waktu dan kecepatan dinyatakan pada persamaan 2.26

$$v = {}^{S}/_{t} \quad (2.26)$$

Dengan :

v = kecepatan rata-rata (*m/s*)
S = Perpindahan (*m*)
t = Waktu (*s*)

2.2.6 Pengiriman Data Melalui Komunikasi *Client Server*

Pada penelitian oleh *celine* (2004) menyatakan bahwa ketika *quadcopter* aktif, secara otomatis akan membuat sebuah *hotspot wifi*. Dengan begitu client dapat tersambung dan dapat melakukan komunikasi berupa pengiriman data. Menggunakan setelan pabrik, SSID *AR.Drone Quadcopter 2.0* dapat diinisialisasi melalui "ardrone2_". *AR.Drone Quadcopter 2.0* dapat berkomunikasi melalui

server DHCP dengan IP 192.168.1.1 dan client pertama dengan IP 192.168.1.2. DHCP Server menggunakan 4 buah socket berbeda dengan fungsi berbeda, socket tersebut antara lain adalah UDP 5556, TCP 5555,UDP 5554, UDP 5559. Selain menggunakan DHCP server, *AR.Drone Quadcopter 2.0* juga menggunakan komunikasi dengan FTP server yang akan berjalan pada port 5551. FTP server dapat mengambil data yang tidak dapat diambil oleh *socket DHCP server* sebagai contoh adalah untuk mengetahui versi *quadcopter*.

2.2.6.1 UDP 5556

Digunakan untuk mengirimkan perintah (*command*) pada *quadcopter*, perintah (*command*) yang bisa dilakukan ialah *command sequencing*, *command structure, watchdog command, the flight mode command, the falt trim command, the flight move command, the control mode command,* dan *the set config value command.*

a. Command Sequencing

Alasan menggunakan command sequence adalah karena untuk mengirim perintah diharuskan mengsertakan command sequence. Karena data yang dikirim diolah secara sequencial, sehingga perintah dapat dilakukan secara urut. Perintah pertama yang harus dilakukan adalah mengirim perintah dengan urutan nomer 1. Perintah yang dilakukan harus melakuka *increment* satu kali, dan tidak boleh ada celah sedikitpun pada urutan. Perintah yang yang tidak sesuai akan otomatis diabaikan.

b. Sequence Structure

Untuk menuliskan perintah, harus diawali dengan “AT*” lalu dikuti dengan nama unik dan simbol sama dengan. Setelah sama dengan diikuti dengan parameter yang diapit oleh kurung siku(*brackets*). Untuk mengakhiri perintah harus menggunakan karakter “\r”.

2.2.6.2 UDP 5554

Socket UDP 5554 digunakan untuk menampilkan data kondisi navigasi quadcopter. Navdata bekerja dengan menentukan antrian data pada port UDP 5554. Struktur penulisan data navigasi pertama adalah dengan cara membawa *section header*, kedua adalah dengan cara mengirimkan *section* jenis lain, terakhir adalah bahwa terdapat nilai *checksum* pada 8 bit terakhir yang dikirim, dengan begitu integritas data dapat dicek.

2.2.6.3 UDP 5559

Data konfigurasi direpresentasikan melalui konfigurasi terakhir yang dilakukan pada *quadcopter*. Hal itu dapat ditentukan menggunakan port data konfigurasi yaitu UDP 5559. Ketika pengguna menginginkan data konfigurasi maka konfigurasi sebelumnya dapat ditentukan. Cara konfigurasi port UDP 5559 dapat ditentukan sebagai *multiline string*. Tiap barisnya dapat menampung satu masukan konfigurasi. *Key* dan *Value* dipecah menjadi bagian yang sama satu sama lain. *Syntax* konfigurasi data pada port UDP 5559 adalah [`KEY`] = [`VALUE`].

2.2.6.4 TCP 5555

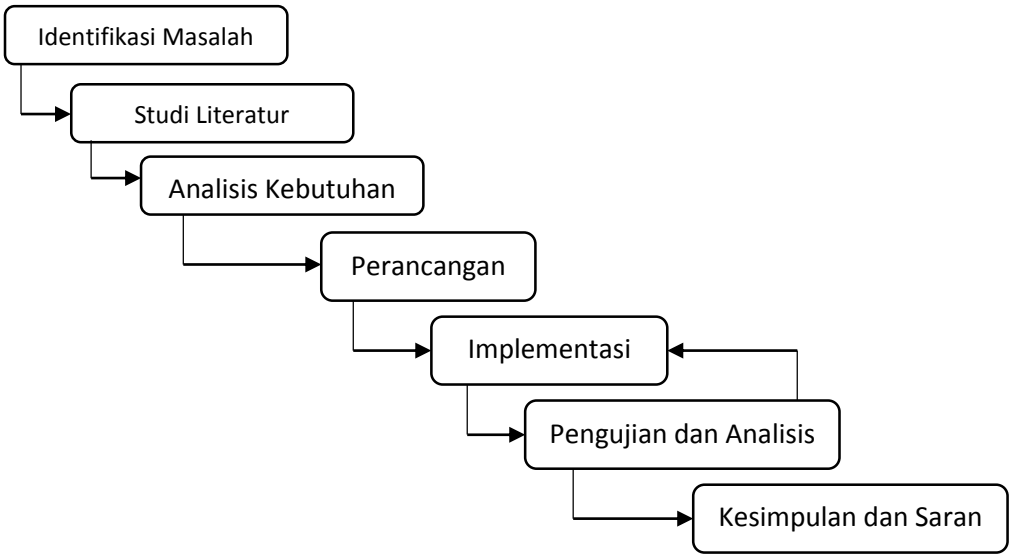
Data *video streaming* dapat dimunculkan melalui port UDP 5555 atau TCP 5555. Karena sistem akan menggunakan *Parrot AR.Drone 2.0* maka port yang digunakan adalah TCP 5555, karena UDP 5555 hanya digunakan untuk *Parrot AR.Drone 1.0*. Agar mendapat data secara *real-time* maka diperlukan *Keep-alive packets*.

2.2.6.5 FTP Server

FTP Server pada quadcopter menggunakan port 5551. Dengan tidak diketahui data dapat diambil menggunakan server ini, dan file index akan secara otomatis tidak diaktifkan. Apabila pada *DHCP sever* menggunakan 4 port berbeda dengan fungsi berbeda, maka pada *FTP server* dapat digunakan untuk semua fungsi, dan bahkan dapat melakukan fitur yang tidak dapat dilakukan pada *DHCP Server*.

BAB 3 METODOLOGI

Pada bab ini akan membahas tentang tahapan penelitian mulai dari Analisis Kebuutuhan, Perancangan, Implementasi, Pengujian, dan Kesimpulan, dengan digram pada gambar 3.1.



Gambar 3.1 Alur Metodologi

Gambar 3.1 merupakan diagram alir yang menujukkan urutan dalam melakukan penelitian. Dimulai dari Identifikasi masalah, Studi literature, analisis kebutuhan, perancangan sistem, implementasi, pengujian dan analisis, serta kesimpulan dan saran. Pada bagian Identifikasi masalah dilakukan observasi mengenai sistem yang akan dibuat, selanjutnya pada tahap studi literatur dilakukakan penguasaan teori yang mendukung tercapainya tujuan sistem. Pada tahap analisis kebutuhan dilakukan pengumpulan data mengenai parameter yang digunakan, pada tahap perancangan dibahas mengenai pemodelan sistem, dan pada tahap implementasi dilakukan integrasi antara *hardware* dan *software*. Pada tahap pengujian dan analisis akan dilihat keberhasilan sistem, apabila keberhasilan masih rendah maka kembali pada tahap implementasi. Dan pada kesimpulan dan saran dilakukan penarikan kesimpulan, dan terdapat saran untuk sistem yang dibuat.

3.1 Identifikasi Masalah

Pada tahap ini dilakukan observasi mengenai masalah yang akan di angkat. Pada tahap ini dibahas mengenai bagaimana masalah terjadi, cara menyelesaikan masalah dengan menggunakan metode yang akan digunakan.

3.2 Studi Literatur

Pada tahap ini dilakukan pendalaman teori mengenai masalah yang diangkat. Pada tahap studi literatur ini juga akan dibahas menganai parameter yang digunakan serta bagaimana hubungannya dengan sistem yang akan dibahas. Pada tahap ini dibahas mengenai teori seperti quadrotor, elektrokimia, kelistrikan dan komunikasi *client* dan *server.*

3.3 Analisis Kebutuhan

Dalam analisis kebutuhan dilakukan identifikasi tentang kebutuhan yang menyangkut sistem, untuk menunjang keberhasilan sistem agar dapat memenuhi tujuan yang akan dicapai. Dalam analisis kebutuhan dibahas mengenai kebutuhan sistem yang mencakup kebutuhan *hardware* dan *software*, kebutuhan komunikasi dan kebutuhan antarmuka. Kemudian dibahas kebutuhan fungsional, serta kebutuhan non-fungsional yang meliputi lingkungan operasi dan asumsi ketetergantungan.

3.4 Perancangan

Dalam perancangan ini akan dibuat rancangan yang terstruktur. Dimana perancangan itu meliputi komunikasi sistem, perancangan pengendalian *quadcopter*, perancangan pengiriman data, perancangan estimasi perpindahan dan waktu, dan perancangan antarmuka web.

3.5 Implementasi

Dalam bagian implementasi, sistem yang dirancang akan diterapkan. Tahap pertama yang dilakukan dalam tahap implementasi adalah pembuatan kode program yang digunakan agar *quadcopter* dan komputer dapat berkomunikasi. Tahap kedua adalah pembuatan kode program yang digunakan agar *quadcopter* dapat melakukan operasi terbang. Pada tahap ketiga, dibuat kode program yang digunakan agar *quadcopter* dapat megirimkan data navigasi menuju komputer. Tahap keempat adalah pembuatan kode program yang dapat mengestimasi waktu terbang dan mengestimasi perpindahan yang mampu ditempuh oleh quadcopter. Dan pada tahap terakhir dibuat kode program tampilan antarmuka web dan cara pengiriman data antara *server node js*, *web socket*, dan *Jquery*.

3.6 Pengujian dan Analisis

Dalam bagian pengujian akan dilihat seberapa besar keberhasilan sistem yang telah diimplementasi. Pada tahap analisis akan dilakukan pengamatan pola-pola yang terjadi. Apabila terjadi kesalahan atau *error* yang cukup tinggi akan dilakukan analisis dan perbaikan sistem. Sehingga didapat tingkat keberhasilan yang tinggi, dan *error* yang rendah. Dalam pengujian akan dilakukan pengujian untuk melihat seberapa tepat pengendailian menggunakan *keyboard*, pengujian untuk melihat akurasi estimasi waktu, dan pengujian untuk melihat ketepatan estimasi perpindahan yang digunakan pada *quadcopter*. Sedangkan pada tahap analisis membahas hasil dari pengujian ketepatan gerakan, analisis hasil pengujian akurasi estimasi waktu, dan analisis analisis hasil pengujian estimasi perpindahan.

3.7 Kesimpulan dan Saran

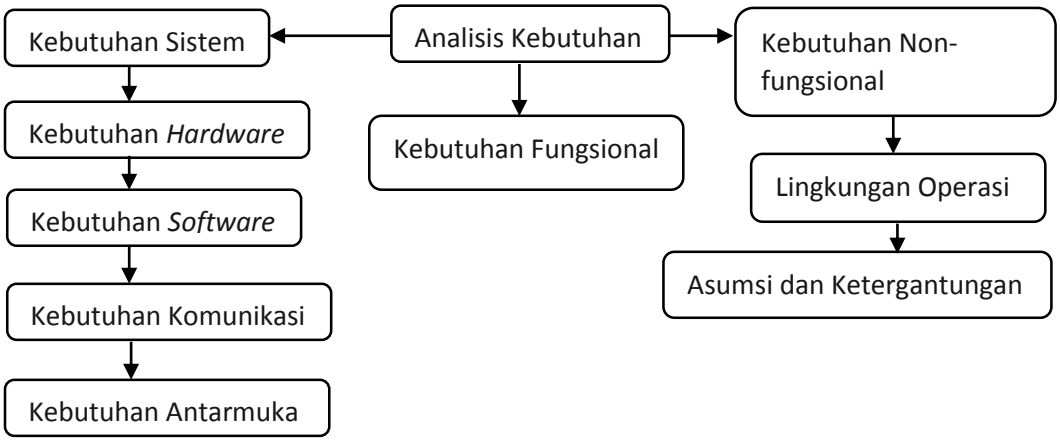
Dalam kesimpulan dan saran akan dibahas mengenai, hasil dari penelitian secara singkat, dengan menganalisis data, memperhatikan pola yang terjadi, serta mendapatkan presentase keberhasilan serta presentase kesalahan. Dalam saran akan dibahas mengenai bagaimana sistem dapat dikembangkan lebih baik lagi melihat dari kekurangan serta kemungkinan bahwa penelitian akan lebih baik jika dikembangkan dengan metode yang disarankan.

BAB 4 ANALISIS KEBUTUHAN

Analisis kebutuhan sistem memliki tujuan untuk menganalisis kebutuhan apa saja yang dibutuhkan untuk membangun sistem. Dalam menganalisis kebutuhan sistem akan membahas kebutuhan pengguna dan kebutuhan sistem itu sendiri. Analisis kebutuhan pengguna memiliki tujuan untuk memudahkan proses *monitoring* agar pengguna yang awam dapat mengetahui status, indikator, serta memahami waktu estimasi dan perpindahan estimasi yang dapat dilakukan oleh *quadcopter*.

Gambar 4.1 Gambaran Umum Sistem

Gambar 4.1 menjelaskan tentang gambaran umum sistem yang dikembangkan. Urutan proses ditunjukkan oleh lingkaran biru dengan nomer. Pertama pengguna yang ditunjukkan dengan lingkaran dengan nomer 1 melakukan konfigurasi pada komputer/*laptop* yang ditunjukkan pada lingkaran nomer 2. Setelah melakukan konfigurasi dan sambungan dengan menggunakan wifi, maka jalur komunikasi antara laptop dan *quadcopter* yang ditunjukkan pada lingkaran nomer 3 sudah terbentuk. Koneksi dan konfigurasi yang telah dilakukan, menjadikan pengguna dapat melakukan operasi terbang, dan ketika *quadcopter* sedang beroperasi, antarmuka *web* yang ditunjukkan pada nomer 4 akan menujukkan status baterai, data navigasi, serta menujukkan estimasi waktu dan perpindahan yang bisa dilakukan oleh *quadcopter*. Maka untuk dapat merealisasikan sistem, diperlukan analisis kebutuhan sesuai dengan gambar 4.2



Gambar 4.2 Diagram Analisis Kebutuhan

4.1 Kebutuhan Sistem

Dalam Kebutuhan antar muka sistem dikelompokkan lagi kedalam tiga kategori agar lebih mudah. Yang pertama adalah kebutuhan antarmuka perngakat keras, yang kedua adalah kebutuhan antarmuka perangkat lunak dan yang ketiga adalah kebutuhan komunikasi.

4.1.1 Kebutuhan Perangkat Keras

Kebutuhan perangkat keras yang diperlukan dalam sistem ini ialah :

4.1.1.1 Parrot Ar.Drone 2.0

Merupakan helicopter *quadcopter* yang dikontrol melalui *remote control* yang dikembangkan oleh *French Company Parrot*. Didukung dengan kamera 720P sangat memungkinkan untuk mengambil gambar secara *HD* dengan *range* seluas 50 meter. *Parrot AR.Drone 2.0* dilengkapi dengan baterai berjenis *Lithium-ion Polymer* dengan tegang sebesar 11.1 volt dan kapasitas sebesar 1500 mAh. Baterai akan mendukung *quadcopter* untuk dapat terbang selama kurang lebih 12 menit.

Untuk mendukung kinerja yang lebih baik *Parrot AR.Drone 2.0* dilengkapi dengan sensor yang canggih. *Parrot AR.Drone 2.0* mendukung kontrol otomatis yang dapat membuat *quadcopter* dapat terbang dan mendarat secara vertikal. Dengan sistem yang mudah dikontrol dan dengan tampilan yang *user-friendly* pada *smartphone* sehingga para pemula tidak akan merasa kesulitan untuk menerbangkannya. Spesifikasi teknis pada *Parrot AR.Drone 2.0* adalah ditunjukkan pada tabel sebagai berikut:

Tabel 4.1 Spesifikasi *Parrot Ar.Drone 2.0*

HD Video	Struktur	*Home*	*Support*	Dimensi dan berat
Live Streaming pada *tablet pc* dan *smart phone*	Desain dengan *figure acrobatic* paling banyak	Baterai *LiPo* 1000 mA/H yang dapat diisi ulang	1 GHz 32-*bit processor*	Badan *indoor* dengan dimensi : 52 x 52 x 11 cm, 420 gr.
Camera 720P HD, 30 fps	Pusat inersia yang aman dari getaran mesin	Dapat melakuka manuver performansi tinggi	1 GB DDR2 RAM pada 200 MHz	Hull *indoor* dimensi : 32 x 25 x 11 cm, 380 gr.
Lebar sudut lensa sebesar 92˚	*Polypropylene* yang di perluas	*8 MIPS AVR Processor* tiap motor	Wi-Fi	
Penyimpanan video pada *USB*	Tabung Karbon *Fiber*	4 *inrunner motor brushless*, 14.5 *watt* daya dan kecepatan 5000 rpm	Sensor *Accelerator* 3 sumbu dengan akurasi +/- 50 mg	

Encoding Profile H264	*Nylon plastic* dengan kualitas tinggi sebesar 30 %	Pelumasan *bearing ball* otomatis	Sensor *gyroscope* 3 sumbu dengan akurasi 2000˚/ detik	
Streaming video dengan latensi yang rendah	Bungkus *nano-hydrophobic* pada sensor ultasonik	Gigi pada baling-baling besi yang diperkuat	Sensor tekanan dengan akurasi 10 *Pascal* (800 *cm DPL*)	
Pengambilan gambar dengan format JPEG	Secara penuh bagian mesin dapat diperbaiki	Perdam *Gear Nylatron* pada baling- baling	*Vertical QVGA* 60 fps *camera*, *Sensor magnetometer* akurasi 6˚	
		Miniatur bola *bearing*	Magnet bumi langka	
		Sensor Ultrasonic untuk pengukuran tinggi	*Emergency Software Shutdown*, Motor yang dapat di program ulang	
		OS Linux 2.6.32, USB 2.0 high speed	Perangkat elektronik tahan air	

Sumber: parrot.com (2017)

Tujuan utama dari sistem adalah dapat memonioring kinerja *quadcopter* saat melalui sisa energi dan daya baterai, maka perlu menganalisis perangkat penting apa saja dalam *quadcopter* yang membutuhkan energy listrik untuk dapat beroperasi. Berikut ini adalah daftar komponen utama yang memerlukan energy listrik untuk dapat beroperasi :

Gambar 4.3 Letak sensor *AR.Drone Quadcopter*

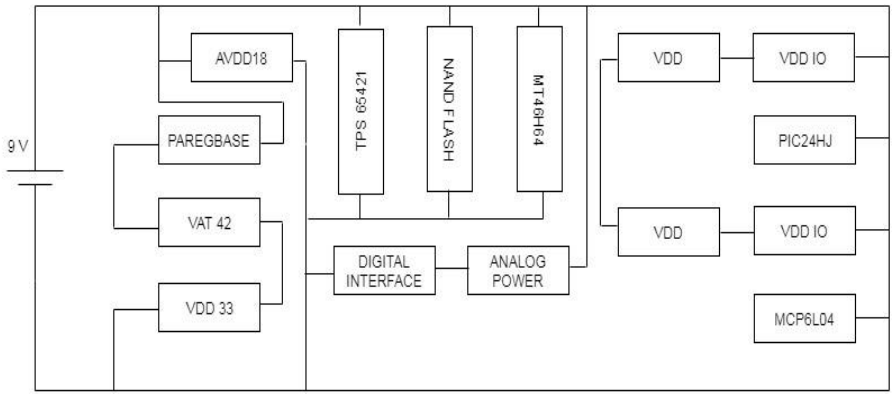
Sumber: parrot.com (2017)

Gambar 4.1 menunjukkan letak letak sensor yang terdapat pada AR.Drone *Quadcopter* 2.0, baterai yang semula memiliki kapasitas 1000 mAh di-*upgrade* menjadi berkapasitas 1500 mAh, penambahan, sensor *magnetometer*, sensor tekanan, serta kamera yang mendukung pengambilan gambar dengan kualitas HD sehingga membuat pengoperasiannya semakin mudah, stabil, serta pengambilan gambar yang jernih. Sensor tak hanya digunakan untuk melihat data pergerakan dari *quadcopter* saja, namun juga dapat dimanfaatkan untuk mendukung sistem yang lebih stabil. Umumnya untuk pemanfaatan kestabilan *quadcopter* digunakan *sensor*, *ultrasonic*, *accelerometer*, *gyroscope*, dan magnetometer(*compass*). Untuk lebih jelasnya, daftar sensor ditunjukkan pada tabel 4.2

Tabel 4.2 Komponen sensor, tranduser dan pemroses *Parrot AR.Drone*

Vendor	***Type***	**Fungsi**
Atheros	*AR6103*	*Wifi Module*
Asahi Kasei	*AK8975C*	*Compass*
Bosch	*BMA150*	*Accelerometer*
Bosch	*BMP180*	*Pressure Sensor*
Microchip	*MCP6L01*	*Ultrasonic Sensor*
Microchip	*PIC24HJ*	*Microcontrollers*
Micron	*MT64H64*	*DDR SDRAM*
Numonix Micron	-	*SLC NAND Flash Memory - 128 MB*
Invansense	*IMU-3000*	*3-axis Gyro & Motion Processor*
Texas Instrument	*TPS65921B*	*Power Management + USB*

Sumber: Teardown.com (2016)



Gambar 4.4 Diagram *Block* Komponen *AR.Drone Quadcopter*

Sumber: Teardown.com (2016)

Gambar 4.2 menunjukkan komponen utama pada *AR.Drone quadcopter*. Pada lampiran komponen utama *AR.Drone* ditunjukkan besaran umum yang dibutuhkan oleh komponen yang terdapat dalam AR.Drone *quadcopter*.

4.1.1.2 Parrot Baterry LiPo

Baterai Lipo atau *Lithium-ion Polymer* adalah baterai yang menggunakan lapisan film yang sangat tipis sebagai komponen yang dapat menyimpan daya,

berbeda dengan polimer kering yang terdapat pada aki kering atau baterai. Dengan Metode penumpuka film yang merupakan tumpukan anoda(positif) dan katoda(negatif) dapat mengoptimalkan daya apabila menggunakan baterai ini. Pada *Parrot* AR.Drone menggunakan baterai dengan spesifikasi sebagai berikut ini :

Tabel 4.3 spesifikasi sumber listrik pada *AR.Drone 2.0*

Kapasitas baterai	1500mAh
Daya dalam Jam	16.6Wh
Durasi Baterai	12 menit
Tegangan	11.1 V
Jumlah *Cell*	3
Discharge Rate	10C
Internal Resistance	752 Ω

Sumber: parrot.com (2017)

Tabel 4.3 spesifikasi sumber listrik pada *Ar.Drone 2.0* merupakan informasi umum yang perlu diketaui oleh pengguna. Dari informasi tersebut dapat digunakan untuk mengetahui seberapa tahan baterai jika digunakan saat *quadcopter* terbang. Dan fungsi lain yang dapat diambil ketika mengetahui informasi tentang baterai adalah dapat mencegah hal – hal yang merugikan pengguna sendiri seperti *over charge* dan *over discharge* yang dapat mengurangi umur baterai.

4.1.2. Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak yang diperlukan dalam pembuatan sistem ini antara lain adalah :

4.1.2.1 Node JS

Node.js merupakan sebuah *platform* perangkat lunak pada sisi *server* dan aplikasi jaringan. Pemrograman pada Node.JS menggunakan bahasa JavaScript dan dapat dijalankan pada sistem operasi *Windows, Mac OS X,* dan *Linux* tanpa perubahan kode program. Node.JS memiliki *library server* HTTP sendiri sehingga memungkinkan untuk menjalankan *web server* tanpa menggunakan program *server web* khusus.

Dalam sistem yang dikembangkan ini *Node JS* digunakan untuk menjembatani komunikasi antara *quadcopter* dengan komputer. Melalui bahasa pemrograman *Node JS* yang biasa digunakan untuk membangun sebuah *web*, antarmuka *software* yang dibangun adalah antarmuka berbasis *Web*. Dimana *Node JS* berjalan pada sisi Server yang akan menavigasi jalannya program, sedangakan disisi *Client* menggunakan *HTML*, *CSS*, dan *Jquery* sebagai tampilannya.

4.1.2.2 FFMPEG

FFMPEG adalah sebuah *software* sumber terbuka yang mampumeproses multimedia dalam banyak format. *FFMPEG* dapat mengkonversi, merekam, dan dapat membantu dalam proses *streaming audio* dan *video*. *FFMPEG* memanfaatkan *CLI*(*Command Line Interface*) dan tidak menggunakan tampilan berupa *GUI*(*Graphical User Interface*) seperti aplikasi multimedia pada umumnya, namun fitur yang ditawarkan serta fungsi dari *FFMPEG* sendiri lebih kompleks dan dapat digunakan dalam banyak format *audio* maupun *video*.

Dalam *Monitoring* AR.Drone *Streaming* kamera dibutuhkan untuk melihat keadaan lingkungan ketika *quadcopter* sedang terbang, serta kamera *quadcopter* juga salah satu perangkat elektronik yang membutuhkan energi baterai agar dapat beroperasi. *Parrot* AR.Drone mendukung penggunaan dua buah kamera, yaitu kamera depan dan kamera bawah. Fungsi dari *FFMPEG* untuk kamera AR.Drone adalah sebagai fasilitator yang menghubungkan antara kamera *AR.Drone* sehingga dapat ditampilkan pada komputer.

4.1.2.3. Sublime

Sublime merupakan sebuah aplikasi editor yang digunakan untuk menuliskan kode program yang digunakan. *Sublime* dapat digunakan untuk menuliskan segala jenis bahasa pemrograman berbasis teks.

4.1.2.4 *Web Browser*

Untuk memfasilitasi antarmuka antara *quadcopter*, komputer, dan pengguna, digunakan *web browser*. Dengan bahasa pemrograman *Javascript* yang mendukung pengambilan data dan umum digunakan dalam aplikasi perangkat lunak *web*, maka digunakan aplikasi *web browser* untuk menampilkan sistem *monitoring* energi baterai *Parrot AR.Drone.*

4.1.3 Kebutuhan Komunikasi

Untuk melakukan komunikasi antara komputer dengan *quadcopter* digunkan protokol *UDP* dan *TCP*. Dimana komunikasi awal adalah dengan menyambungkan *quadcopter* dengan komputer menggunaka *Wi-Fi*, setelah itu data akan dikomunikasikan melalui protokol tersebut.

4.1.4 Kebutuhan Antarmuka

Antarmuka digunakan untuk menjembatani komunikasi antara pengguna dengan sistem. Komunikasi antara *quadcopter* dan komputer yang dijembatani oleh *Javascript* yang merupakan bahasa yang sering digunakan untuk membangun web, maka dalam sistem *monitoring* baterai *Parrot AR.Drone* ini akan menggunakan antarmuka berupa *web browser* dimana pengguna dapat mengetahui estimasi waktu dan perpindahan, dapat melihat lingkungan sekitar, dapat mengetahui data navigasi, dan dapat mengetahui indikator *quadcopter*.

4.2 Kebutuhan Fungsional

Kebutuhan fungsional adalah kebutuhan yang mampu memberikan pernyataan yang mampu menunjukkan bahwa sistem akan berjalan sesuai dengan

fungsinya, serta dapat menghasilkan ouput seperti yang diharapkan. Kebutuhan Fungsional tersebut antara lain adalah :

1. Mampu memantau kinerja baterai *quadcopter* melalui web browser.
2. Sistem mampu memberikan layanan untuk mengendalikan *quadcopter* melalui web browser, seperti melakukan take-off, landing, maju, mundur, keatas, dan kebawah.
3. Sistem menujukkan data naviagasi, indikator berupa kapasitas baterai, ketinggian, temperature *quadcopter*, dan waktu aktif melaui tampilan berupa grafik, *progress bar*, dan angka
4. Dapat mengetahui lingkungan sekitar saat *quadcopter* terbang.
5. Dapat menampilkan estimasi perpindahan dan waktu berdasarakan sisa kapasitas baterai.

4.3 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional akan membahas tentang apa saja yang akan menjadi batasan dalam perancangan sistem. Kebutuhan non-fungsional tersebut antara lain adalah :

Tabel 4.4 Kebutuhan non-fungsional

Property	*Measuring*
Robustness	- Ketahanan *quadcopter* mengenai seberapa lama waktu yang dapat dilakukan *quadcopter* ketika melakukan operasi pada saat energi baterai menurun
Portability	- Kemampuan untuk berpindah tempat dari satu tempat ke tempat lainnya pada saat energi baterai menurun

4.3.1 Lingkungan Operasi

Analisis lingkungan operasi ditujukan agar menciptakan suasana yang aman antara pengguna dengan *quadcopter*, baling-baling yang berputar sangat cepat dapat mencelakai diri sendiri dan orang lain, serta lingkungan yang terlalu sempit atau terlalu ramai juga dapat merusak *quadcopter* itu sendiri, sehingga diperlukan posisi yang tepat serta lingkungan yang cukup luas dan tinggi untuk ruang tertutup(indor) dan ruangan yang tidak terlalu ramai/ tidak terdapat ranting yang berdekatan. Adapun persyaratannya lingkungan yang dapat digunakan adalah sebagai berikut :

1. Berada pada ruangan yang terdapat sumber daya listrik untuk kebutuhan listrik pada laptop/komputer
2. Pengoperasian tidak berada diruangan yang penuh perabotan.

3. Ketika diluar ruangan memperkirakan kecepatan angin, jika angin sedang maka lebih baik dapat menggunakan *outdoor hull*.

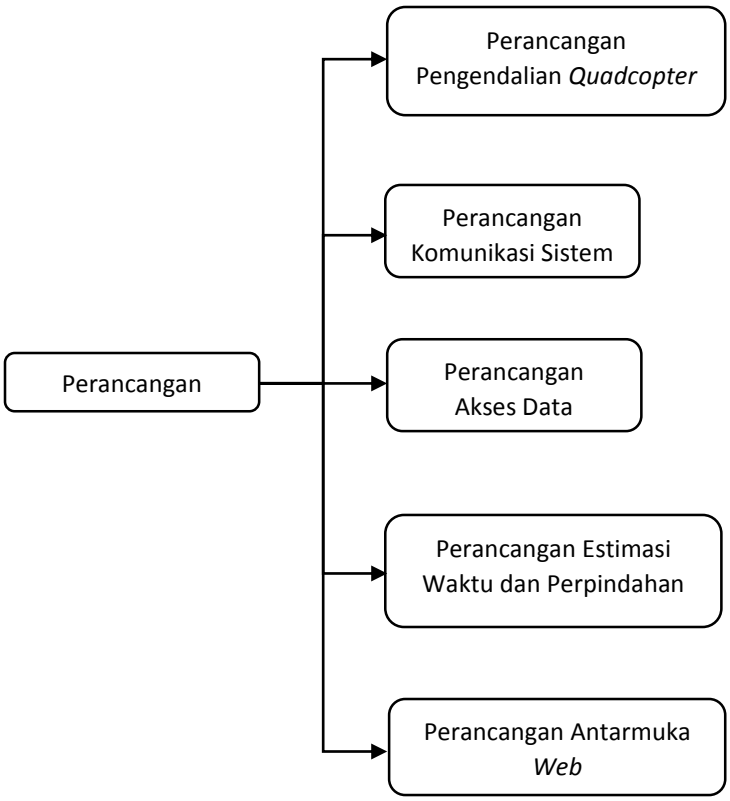
4.3.2 Asumsi dan Ketergantungan

Dalam sistem selalu ada Ketergantungan terhadap sesuatu, untuk perangkat elektris hal yang paling umum dibutuhkan adalah daya. Dalam asumsi dan ketergantungan akan dibahas mengenai tentang apa saja yang menjadi hal yang diperlukan dalam menjalankan sebuah sistem. Yang mana hal tersebut antara lain adalah :

1. Baterai AR.Drone harus terisi paling tidak diatas 50%
2. Harus melakukan sambungan melalui *wifi* agar komputer dapat terhubung dengan *quadcopter* untuk dapat diterbangkan.
3. Untuk menjalan program, server Node JS harus dipanggil terlebih dahulu melalui Node JS command Prompt.
4. Harus membuka web browser, serta memanggil local host pada url dan port yang digunakan oleh server.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Pada bab ini akan dibahas mengenai perancangan yang akan digunakan sebagai acuhan dan pertimbangan untuk mengimplementasikan sistem, setelah perancangan sistem selesai maka akan masuk pada tahap implementasi dimana komponen berupa *hardware* dan *software* akan diintegrasikan sehingga membentuk suatu sistem yang sesuai dengan tujuan sistem ini dibuat. Diagram pada gambar 5.1 akan menunjukkan urutan proses yang dilakukan untuk merancang serta mengimplementasikan sistem.

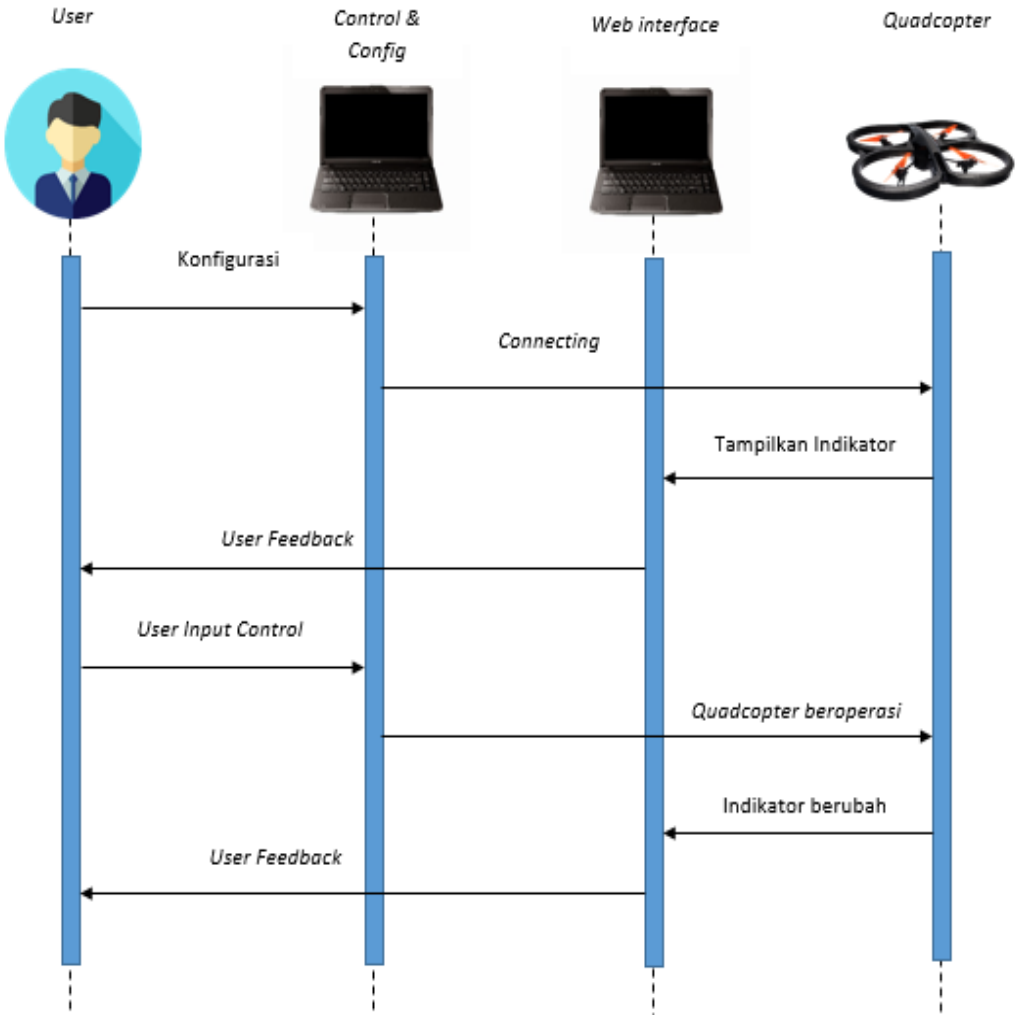


Gambar 5.1 Alur Perancangan Sistem

5.1 Perancangan Sistem

5.1.1 Perancangan Komunikasi Sistem

Parrot AR.Drone 2.0 melakukan koneksi dengan perangkat seperti *smartphone* atau *laptop* menggunakan sambungan *wifi*. *Parrot AR.Drone* memiliki alamat IP *default* yaitu 192.168.1.1 yang bisa diakses menggunakan komputer/*laptop*, *tablet pc*, dan *smartphone*. Komunikasi sistem melibatkan *laptop/pc*, *server* yang dibangun dengan bahasa pemrogramana *Javascript*, antarmuka aplikasi *web* yang dibanguna menggunakan *HTML, CSS*, dan *Jquery*.

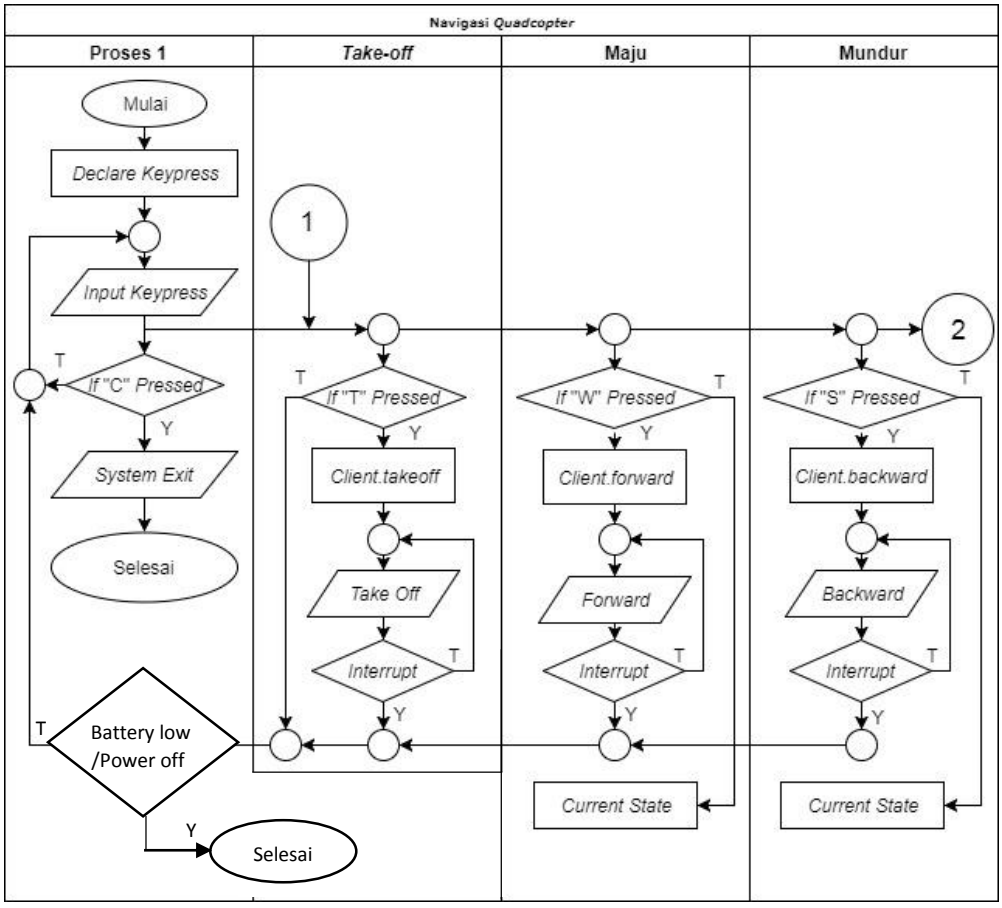


Gambar 5.2 Perancangan Komunikasi

Gambar 5.2 menjelaskan alur komunikasi yang diimplementasikan. Komunikasi antara *quadcopter* dan komputer dilakukan dengan menggunakan wifi. Program utama yaitu *node js server* akan melakukan komunikasi dengan paket-paket *node js* yang digunakan, contoh paket yang dipanggil adalah *Node AR.Drone*. Untuk dapat memanggil paket-paket yang lain digunakan *express* yang digunakan untuk *routing*. Dan terdapat *Socket IO* yang digunakan untuk mengirimkan data yang diproses dalam *node js server* sehingga mampu ditampilkan pada antarmuka *web*.

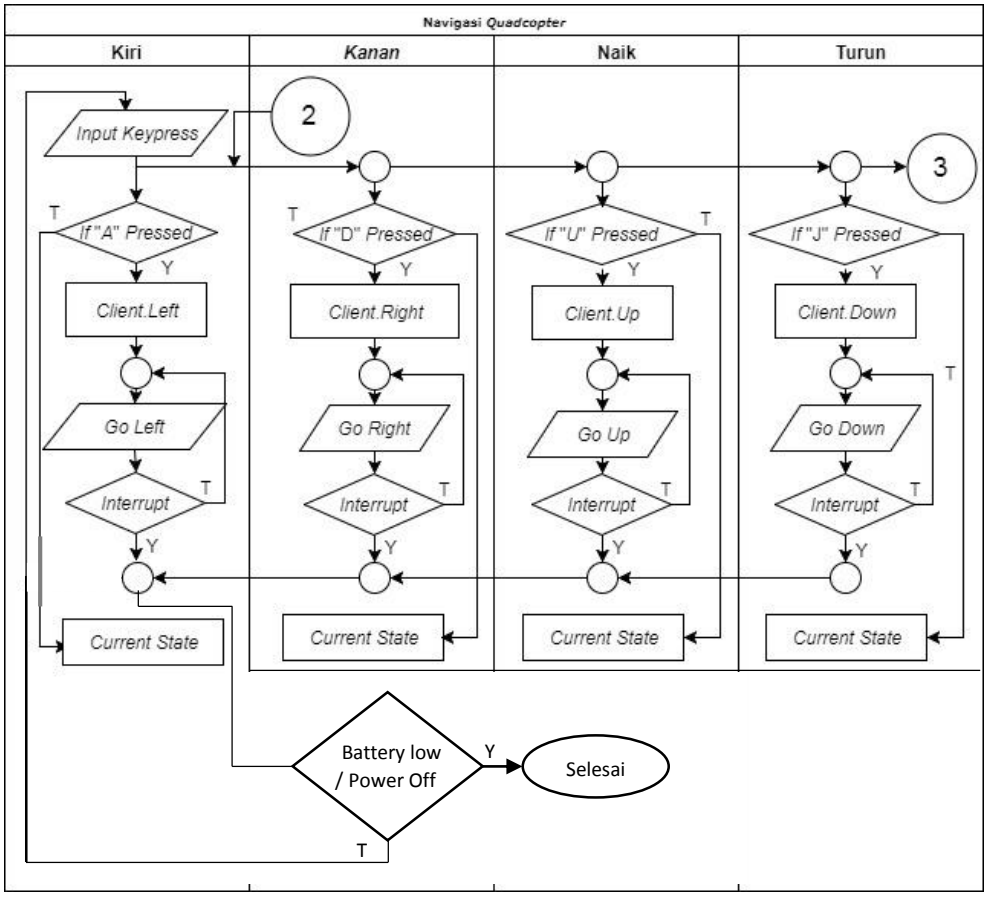
5.1.2 Perancangan Pengendalian *Quadcopter*

Dalam sistem yang dikembangkan ini, pengoperasian adalah hal yang sangat penting, karena tujuan utama dari sistem ini adalah untuk dapat mengestimasi perpindahan dan waktu tempuh dari quadcopter. Tanpa dapat beroperasi atau terbang tentunya kita tidak akan bisa untuk mengestimasi perpindahan dan waktu. Operasi quadcopter yang dilakukan dalam sistem ini adalah dengan menggunakan keyboard laptop. Gerakan *quadcopter* antara lain adalah, maju, mundur, kanan, kiri, atas, bawah, *clockwise* dan *counterclockwise.*



Gambar 5.3 *Flowchart 1* Pengendalian *Quadcopter*

Gambar 5.3 menujukkan rancangan sebagian gerakan pada *AR.Drone Quadcopter*. Tahap pada gambar 5.3 adalah dimulai dari proses 1, *take-off*, maju, dan mundur. Pada proses 1 terdapat proses deklarasi penggunaan *module keypress*, setelah module keypress selesai dideklarasi terdapat kondisi jika C ditekan maka akan keluar dari proses. Operasi *Take-off* dapat diakses ketika kondisi jika tombol T dari *keyboard* ditekan maka akan terdapat proses yang memanggil AT*COMMAND *control*, sehingga *quadcopter* dapat terbang, dan jika kondisi *interrupt* benar maka akan kembali pada input keypress. Sama dengan *take-off* operasi maju akan terjadi jika tombol W dari *keyboard* telah ditekan, AT*COMMAND *control* dipanggil, dan *quadcopter* akan maju, jika interrupt bernilai benar maka akan kembali ke input keypress, dan jika kondisi jika tombol *W* tidak terpenuhi maka *quadcopter* akan tetap pada operasi sebelumnya. Operasi maju akan bekerja dengan cara diinput dari tombol *S keyboard*, sehingga menjalankan AT*COMMAND *control*, dan *quadcopter* akan mundur, jika kondisi interrupt benar maka akan kembali pada input keypress, dan jika kondisi input tombol *S* tidak terpenuhi maka *quadcopter* akan berada pada operasi sebelumnya. Dan pada kolom maju terdapat instruksi yang tersambung, yang akan dilanjutkan pada gambar 5.4.

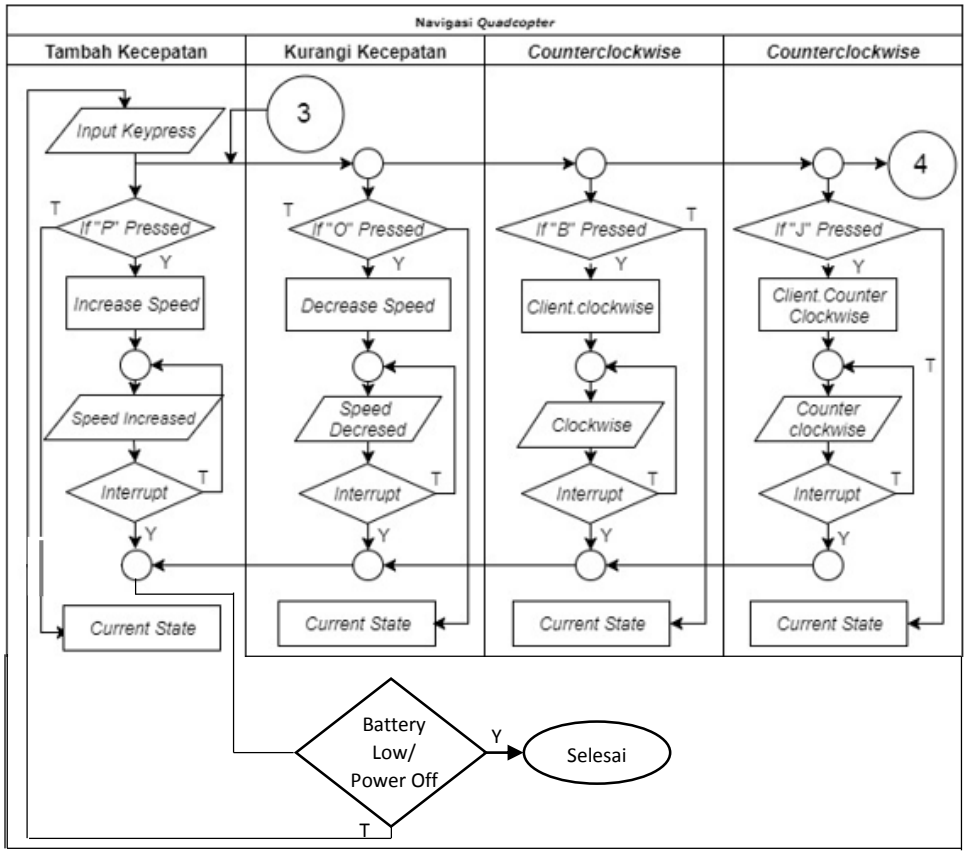


Gambar 5.4 *Flowchart* 2 Pengendalian *Quadcopter*

Gambar 5.4 merupakan proses lanjutan dari proses yang ada pada gambar 5.3. Operasi yang ditunjukkan pada gambar 5.4 adalah gerakan kiri, gerakan kanan, gerakan naik, dan gerakan turun. Operasi gerakan kiri akan terpenuhi apabila tombol "A" pada keyboard ditekan(kondisi terpenuhi), tetapi apabila kondisi tidak memenuhi maka *quadcopter* akan berada pada *state* sebelumnya. Setelah tombol "A" pada keyboard ditekan maka sistem akan memanggil operasi gerakan ke kiri dengan AT*COMMAND *control*, dan *quadcopter* akan terbang kekiri, dan jika kondisi *interrupt* terpenuhi maka akan berada pada input *keypress*, dan jika kondisi *interrupt* tidak terpenuhi maka *quadcopter* akan tetap menjalankan operasi gerakan ke kiri. Operasi gerakan kanan akan terpenuhi apabila tombol huruf "D" pada *keyboard* ditekan, namun jika tidak ditekan maka *quadcopter* akan berada pada *state* sebelumnya. Selanjutnya sistem akan memanggil AT*COMMAND *control* agar *quadcopter* dapat melakukan gerakan ke kanan, dan jika kondisi *interrupt* terpenuhi maka akan kembali pada *input keypress*, dan bila tidak terpenuhi maka *quadcopter* akan melakukan operasi gerakan terbang kanan.

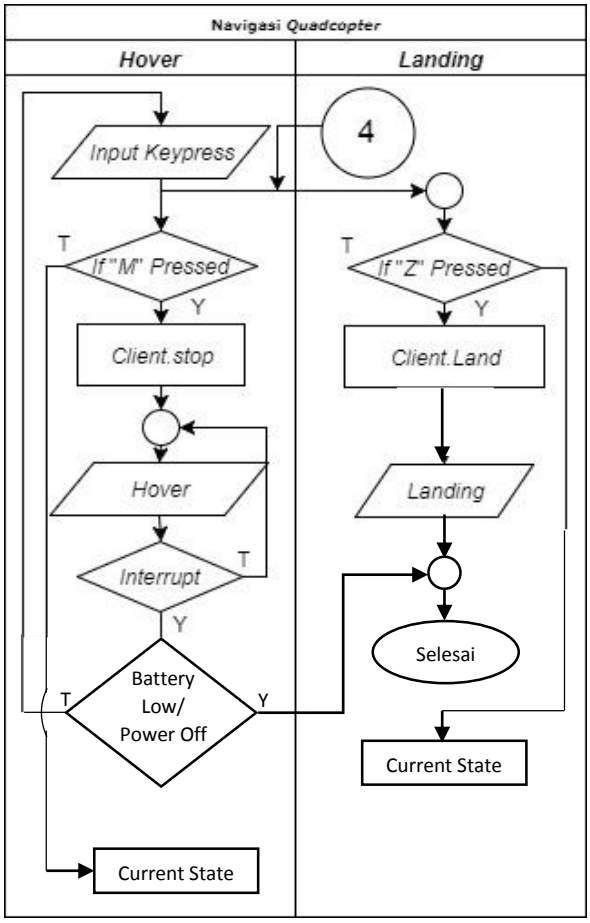
Operasi gerakan terbang naik akan terpenuhi jika tombol huruf "U" pada *keyboard* ditekan dan akan memanggil AT*COMMAND *control* sehingga quadcopter dapat terbang ke atas, jika proses gerakan ke atas di-*interrupt* maka akan masuk kedalam *input keypress*, dan jika tidak maka quadcopter akan tetap pada operasi gerakan terbang ke atas. Operasi gerakan terbang ke bawah akan terpenuhi jika tombol huruf "J" pada keyboard ditekan, sehingga dapat memanggil

AT*COMMAND control. Jika gerakan terbang ke bawah di-*interrupt* maka akan kembali ke input keypress, dan jika tidak maka *qudcopter* akan tetap melakukan gerakan terbang ke bawah. Pada kolom paling kanan terdapat bulatan dengan nomer 3 yang mana proses akan diteruskan pada gambar 5.5



Gambar 5.5 *Flowchart Controller Quadcopter*

Gambar 5.5 adalah proses lanjutan yang telah dijelaskan pada gambar 5.4, selanjutnya akan membahas operasi gerakan *Clockise* dan *counterclockwise*, serta penambahan dan pengurangan input kecepatan yang dapat dilakukan oleh pengguna. Ketika tombol huruf "B" di-*input* kan maka terjadi proses yang di lakukan AT*COMMAND yang menjadikan *quadcopter* dapat terbang dan bergerak clockwise(memutar searah jarum jam), dan jika dilakuka interrupt dengan menekan tombol yang sudah desediakan maka sistem akan masuk kembali kedalam *input keypress*. Operasi gerakan *counterclockwise* akan terjadi jika pengguna menekan tombol N pada *keyboard*, setelah tombol ditekan maka akan masuk kedalam proses yang mamnggil AT*COMMAND sehingga akan menghasilkan keluaran berupa operasi gerakan *quadcopter counterclockwise*(berlawanan arah jarum jam). Pada proses penambahan dan pengurangan kecepatan digunakan tombol *keyboard* P dan O, mekanismenya adalah apabila tombol P ditekan maka input kecepatan akan ditambah sebesar 0.1. Dan untuk pengurungan apabila tombol "O" ditekan maka kecepatan akan dikurangi sebesar 0.1. Lanjutan flowchart dari gambar 5.5 adalah *flowchart* 5.6.



Gambar 5.6 *Flowchart Controller quadcopter*

Gambar 5.6 merupakan lanjutan dari *flowchart* gambar 5.5, yang akan menjelaskan hover dan landing. Untuk gerakan *hover* dimulai dari kondisi jika tombol "M" diinputkan oleh pengguna, maka sistem akan memanggil *AT*COMMAND control* yang akan memproses agar *quadcopter* berhenti melakukan perpindahan. Apabila tmbol "M" tidak ditekan maka *quadcopter* akan tetap melakukan gerakan sebelumnya. Dan apabila setelah proses *quadcopter* berhenti pengguna melakukan interrupt, maka sistem akan kembali pada input keypress. *Landing*/ mendarat akan terpenuhi apabila pengguna menekan tombol Z pada *keyboard*, dan tetap akan memanggil AT*COMMAND yang menyebabkan *quadcopter* mendarat.

Gambar 5.3 hingga 5.6 merupakan flowchart yang digunakan sebagai rancangan yang digunakan untuk mengoperaiskan atau menerbangkan AR.Drone quadcopter. Alasan dipilihnya keyboard sebagai *controller* adalah karena mudah, tidak memerlukan tambahan lain user dapat mengoperasikan quadcopter seperti memainkan sebuah game, karena keys yang digunakan sering juga digunakan dalam *controller game*.

Tabel 5.1 Fungsi *keypress*

Key	Fungsi
C	Untuk menghentikan sistem
T	Untuk *take-off*
Z	Untuk *landing*
W	Untuk terbang maju
A	Untuk terbang ke-arah kiri
S	Untuk mundur
D	Untuk terbang ke arah kanan
N	Untuk gerakan *counterclockwise*
B	Untuk gerakan *clockwise*
M	Untuk menghentikan *quadcopter* / *hover*
O	Untuk mengurangi kecepatan
P	Untuk menambah kecepatan
U	Untuk gerakan naik
J	Untuk gerakan turun

Keypress adalah sebuah paket dari *Node JS* yang dapat digunakan untuk memberikan fungsi *input* dari *keyboard*. *Keypress* akan melakukan *interrupt* kedalam sistem dengan keluarannya adalah gerakan *quadcopter*. Sehingga jika *keypress* yang diset terpanggil, maka akan memanggil *object* gerakan dari *quadcopter*. Dalam *object* gerakan *quadcopter* terdapat fungsi kecepatan yang dapat diubah. Pengubahan kecepatan dapat dilakukan secara langsung dengan menggunakan *keypress*. Terdapat batas dimana jika pengguna terus menerus menekan tombol untuk mengurangi kecepatan, maka nilai kecepatan akan kembali ke *default.* Dan apabila terus menerus menekan tombol untuk menambah kecepatan.

5.1.3 Perancangan Akses Data

Informasi tentang *AR.Drone* seperti posisi, kecepatan, kecepatan putaran mesin, dan indikator lainnya disebut dengan *navigation data* biasa disingkat dengan *navdata* merupakan data yang dikirim kepada aplikasi klien dan akan diterima dalam waktu kurang dari 5 *millisecond*. Data tersebut mengandung informasi status *quadcopter*. *Navdata* digunakan untuk mengirimkan nilai dari sensor dari *quadcopter*. Data berupa angka yang nantinya akan diolah. *Navdata* disimpan kedalam *variable* yang mempunyai tipe data berbeda-beda, tergantung dari besaran dan satuan unit dari sensor. Berikut ini adalah tipe data dan jenis-jenis *navdata* yang diakses dalam sistem ini.

Tabel 5.2 Tipe *Data Navdata*

Property	*Unit*	*Data Type*	*Bits*	*Sign*	*Range*
Battery Capacity	%(from V)	uint32	32	*Unsigned*	0…4,294,967,295
Altitude	Mm	int32	32	*Signed*	-4,294,967,295… 4,294,967,295

Velocity	mm/s	float32	32	*Floating points*	
Acceleration	m/s^2	uint16	16	*Unsigned*	0…65.535
Motor	PWM	uint8	8	*Unsigned*	0…255
Time	Millisecond	uint32	32	*Unsigned*	0…4,294,967,295
Temperature	Celcius	int32	32	*Signed*	

Sumber: Piskroski (2012)

Dari tabel tipe data *navdata* dapat diketahui bahwa tipe data yang digunakan adalah *unsigned integer*, *signed integer*, dan *floating points*. Perbedaan yang dapat dilihat dari ketiga tipe data tersebut adalah *unsigned int* dimulai dari 0, *signed int* dimulai dari angka negatif, dan *floating points* merupakan nilai desimal. Rentang diatas menyatakan banyaknya *bit* yang dikirim, sesuai dengan yang telah dijelaskan diatas bahwa nilai dari *navdata* berbeda-beda sesuai dengan besar dan satuan yang dipkai oleh masing masing data yang di deteksi dan dioleh sensor dan tranduser dari *AR.Drone Quadcopter*.

Komunikasi *Ar.Drone Quadcopter* dengan piranti lainnya dilakukan dengan protocol *TCP* dan *UDP*, yang disediakan port-port yang bisa diakses untuk fungsi yang berbeda. Untuk mengakses *navdata*, digunakan *port* 5554.

Navigation Data Header					*Navigation Data Tags*			*Checksum Block*		
Identifier 32 bit	*State bit mask 32 bit*	*Sequence Number 32 bit*	*AR State Bit Mask 32 bit*		*Tag Identifier 16 bit*	*Length of Section 16 bit*		*Tag 16 bit*	*Size 16 bit*	*Cks data 32 bit*

Gambar 5.7 *Navigation Data Structure*

Sumber: Piskroski (2012)

Gambar 5.7 menjelaskan bagaimana *navdata* dapat dikirimkan. Proses paling awal dari navigasi data adalah *Navigation data structure* yang mempunyai 16 *Byte header* dengan *identifier* 4 *Byte*, *state bit mask* 4 *byte*, dan *AR state bit mask* 4 *Byte*. Untuk dapat diakses, *header* harus selalu bernilai sama dengan $0x55667788_{(8)}$ yang diset kedalam bilangan *oktal*. Jika *header* tidak bernilai sama dengan nilai *header* yang diset maka otomatis data dapat dibuang.

Setelah mengirim *identifier* selanjutnya perlu untuk memanggil *bit mask*, *bit mask* disini digunakan untuk memanggil *navdata*. *State bit mask* bernilai 4 *Byte* atau 32 *bit*. *Bit* 1 hingga 32 akan memuat nilai data, tetapi data ke-13 tidak digunakan. Isi nilai data dari *state bit mask* ditunjukkan pada gambar 5.8.

Flying(0)	*Video Enable(1)*	*Vision Enable(2)*	*Control Algorithm(3)*	*Altitude Control Active(4)*	*User Feedback(5)*	*Control recieve(6)*	*Trimm Recieve(7)*
Trim Running(8)	*Trimm Succeded(9)*	*Navdata Demo Only(10)*	*Navdata bootstrap(11)*	*Motors Down(12)*	*Gyrometers Down(14)*	*Battery Too Low(15)*	*Battery Too High(16)*
Timer Elapsed(17)	*Not Enough Power(18)*	*Angles out of Range(19)*	*Too Much Wind (20)*	*Ultrasonic sensor Deaf (21)*	*Cutout System Detected(22)*	*Pic version Number OK(23)*	*AT Codec Thread On(24)*
Navdata Thread On(25)	*Video Thread On(26)*	*Acquisition Thread On(27)*	*Control Watchdog Delayed(28)*	*Adc Watchdog Delayed(29)*	*Communication Problem Occure(30)*	*Emergency(31)*	

Gambar 5.8 Isi Nilai *State Bitmask*

Sumber: Piskroski (2012)

Langkah setelah *navigation data header* dan *state Bit Mask* adalah *sequence number*. *Sequence number* digunakan untuk mengurutkan data yang dipanggil. Mekanisme *sequence number* adalah *increment* dari angka yang dimulai dari 0. Dan urutan *navigation data structure* yang terakhir adalah *AR bit mask*.

Selanjutnya untuk dapat mengirimkan data perlu dilakukan konfigurasi. Oleh karena itu aliran *navdata* membutuhkan *AT Command*. Dalam *AR.Drone SDK 2.0* telah disiapkan beberapa *AT Command* yang bisa di akses untuk dapat mengkonfigurasi sistem, bila pada control menggunakan *AT*REF* dan *AT*PCMD*, navdata menggunakan *AT*CONFIG* dan *AT*CONFIG_IDS*. Tabel dibawah ini menyjikan dan menjelaskan tentang *AT Command* yang disediakan oleh *AR.Drone Quadcopter*.

Table 5.3 AT Command

AT Command	Arguments	Description
AT*REF	*Input*	*Take off/ landing/emergency*
AT*PCMD	*Flag, roll, pitch, yaw*	*Move the quadcopter*
AT*PCMD_MAG	*Flag, roll, pitch, gaz, yaw, psi, psi accuracy*	*Move the quadcopter*(dengan control mutlak)
AT*FTRIM	-	*Set referensi* awal
AT*CONFIG	*Key, value*	Konfigurasi *AR.Drone 2.0*
AT*CONFIG_IDS	*Session, user application ids*	*Indentifier* untuk *AT*CONFIG command*
AT*COMWDG	-	*Reset* komunikasi *Watchdog*
AT*CALIB	*Device number*	Meminta *AR.Drone* kalibrasi *magnetometer* sehingga harus terbang.

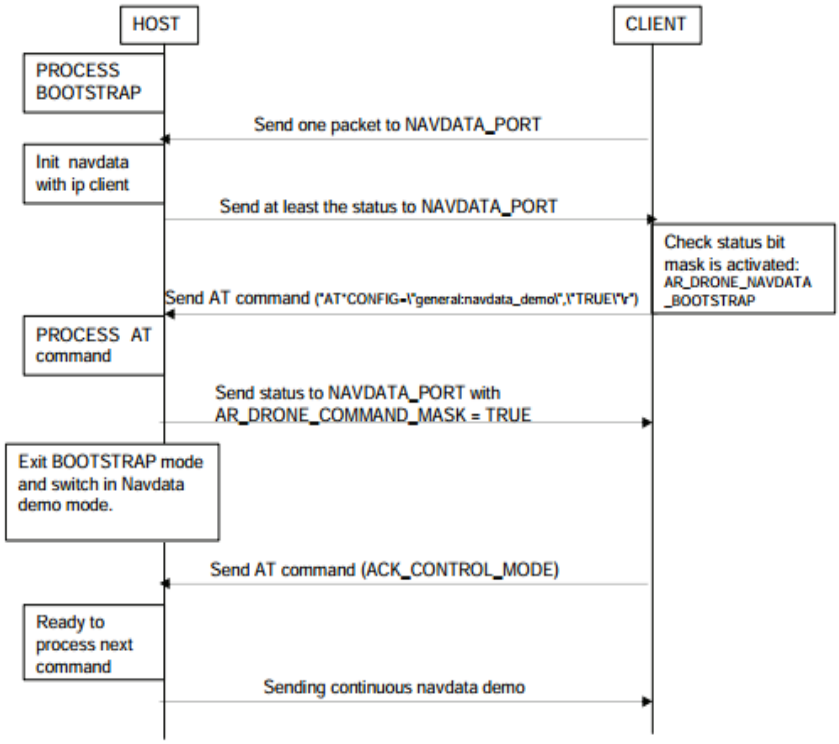
Sumber: Piskroski (2012)

1. AT*CONFIG
   Tujuan utama dari AT*CONFIG adalah untuk mengatur konfigurasi "*option*" dari *quadcopter*. *Option* sendiri adalah sebuah *set* yang berisi informasi yang dibutuhkan yang bisa dikonfigurasikan menggunakan *command* ini. *Syntax* untuk menggunakan *Command* ini adalag sebagaia berikut ini.
   *AT*CONFIG = %d,%s,%s<CR>*
   Argument 1 : Berisi *sequence number*
   Argument 2 : Nama *option(*diantara *double quote)*
   Argument 3 : Nilai dari *option(*diantara *double quotes)*
2. AT*CONFIG_IDS

   Command ini berfunsgi sebagai identifier untuk command AT*CONFIG selanjutnya. *Syntax* dari AT*CONFIG_IDS adalah sebagai berikut.

AT*CONFIG_IDS = %d,%s,%s,%s <CR>
Argument 1 : sequence number
Argument 2 : Session ID terakhir
Argument 3 : User ID terakhir
Argument 4 : ID Aplikasi terakhir

Untuk dapat mengakses *navdata* , paket harus dikirim pada beberapa *bit* pada *NAVDATA port*. Terdapat kasus dimana quadcopter akan berada pada *bootstrap mode* sehingga hanya status dan *sequence number* saja yang akan dikirim, dan *quadcopter* yang akan mengirimkan demo mode. Untuk dapat keluar dari mode *bootstrap* klien harus mengirimkan AT *Command* untuk memodifikasi konfigurasi dari *quadcopte*r. Setelah berhasil keluar dari *mode bootstap* maka sistem akan memerintahkan klien untuk masuk pada mode control, dan bisa mengirimkan data secara terus - menerus.

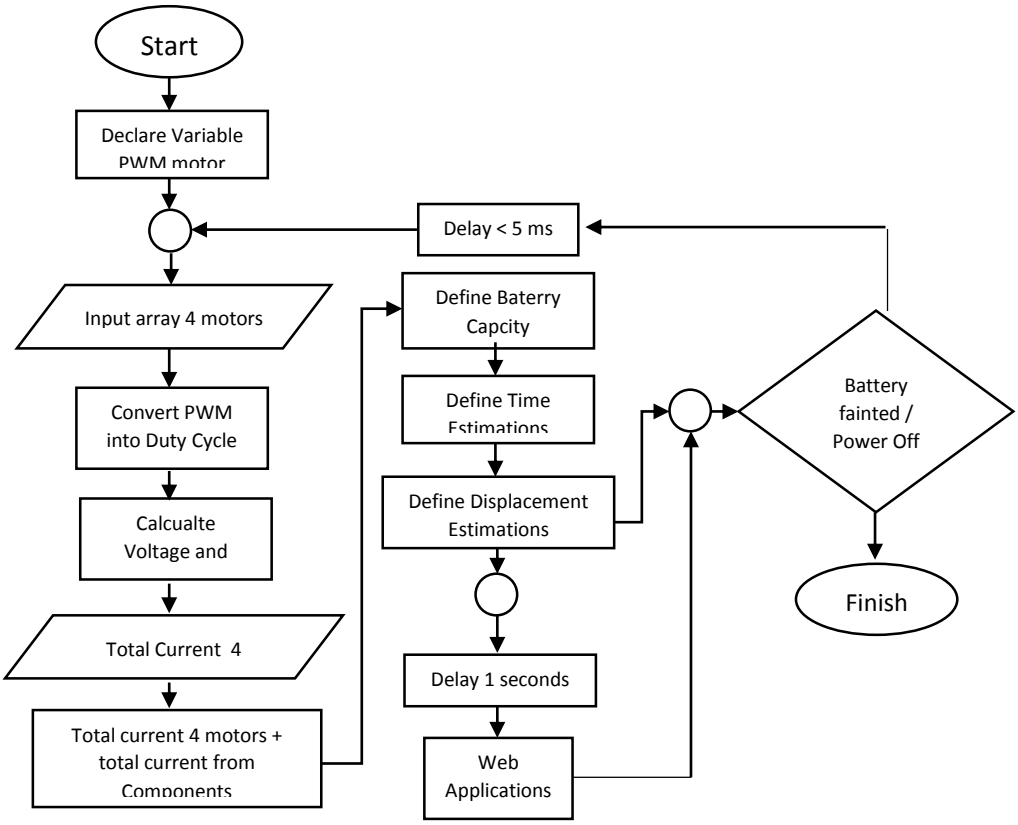


Gambar 5.9 Aliran Pengiriman Data
Sumber: Piskroski (2012)

Gambar 5.9 menunjukkan komunikasi antara *host* dan *client* yang sedang melakukan pertukaran informasi. Untuk dapat mengirimkan status berupa *navdata* dilakukan komunikasi secara bergantian. Hal yang pertama dilakukan oleh *client* adalah meminta *host* agar mengirimkan paket. Kemudian *host* akan menginisialisasi *navdata* menggunakan ip *client*. Setelah data dikirimkan dilakukan *ack* untuk memferivikasi data yang dikirimkan. Jika data telah diverifikasi, maka *host* akan mengirimkan *navdata* dengan konfigurasi demo secara terus -menerus.

5.1.4 Perancangan Estimasi Waktu dan Perpindahan

Melalui persamaan matematis (2.20) dikatahui bahwa untuk dapat mengetahui seberapa lama baterai dapat menyediakan energi bagi perangkat elektronik diperlukan besaran berupa arus dan nilai kapasitas baterai. Sedangkan untuk dapat mengetahui seberapa jauh *quadcopter* dapat terbang, dilakukan pengujian dengan gerakan berbeda dan kecepatan yang berbeda, akan menempuh perpindahan seberapa jauh dengan lama satu menit.



Gambar 5.10 Flowchart Proses Perhitungan Waktu Dan Perpindahan

Pada gambar 5.10 *flowchart* proses perhitungan waktu dan perpindahan. Dimulai dengan mendeklarasikan *variable* yang akan menyimpan nilai *PWM* dari ke-4 motor. Seperti yang telah dijelaskan pada tabel 4.1 dimana daya dari motor adalah 14.5 *watts*, dan tegangan baterai adalah 11.1 *volt*, maka dengan menggunakan persamaan (2.8) tentang daya, maka didapat perhitungan.

$$P = V \times I \tag{5.1}$$

$$14.5 = 11.1 \times I \tag{5.2}$$

Maka $I = 1.3A$ (5.3)

Diketahui bahwa arus yang mampu digunakan oleh sebuah motor *brushless* adalah 1.3 *Ampere*. *Navdata PWM* motor yang diambil merupakan *array*. *Array* dimulai dari index ke-0, sehingga untuk mengambil nilai motor 1 dan

4, perlu mengakses array pada index ke-0 hingga 3. Setelah *array* dapat diakses maka *output* dari proses tersebut adalah nilai pwm yang bernilai maksimal 255. Nilai *PWM* 255 jika diubah menjadi *duty cycle* adalah 100 %. Dari persamaan (2.24) didapat nilai *duty cycle*.

Sebagai contoh dari gambar 2.16 kita dapat mengetahui bahwa *duty cycle* 50% atau ½ *power* memiliki nilai *PWM* 128. Jika nilai *PWM* sebesar 128 disubstitusikan kedalam persamaan (2.24), maka didapatkan

$$Duty\ Cycle = \frac{nPwm \times 100}{255} \quad (5.4)$$

$$128 \times {}^{100}/_{255} = 50\% \quad (5.5)$$

$${}^{50}/_{100} \times 11.1 = 5.55\ V \quad (5.6)$$

Persamaan (2.5) merupakan cara untuk mencari arus listrik menurut hukum *ohm*. Setelah diketahui tegangan dan arus maksimal, maka perlu dicari hambatan, yang mana menggunakan persamaan (2.6).

$$R = {}^{V}/_{I} \quad (5.7)$$

$${}^{11.1}/_{1.3} = 8.5\ \Omega \quad (5.8)$$

Dan apabila menggunakan nilai PWM 128, *duty cycle* 50 %, tegangan 5.55 *volt*, dan hambatan sebesar 8.5 *ohm*, dapat diketahui nilai arus sdengan menggunakan persamaan (2.5).

$$I = {}^{V}/_{R} \quad (5.9)$$

$${}^{5.55}/_{8.5} = 0.6\ Ampere \quad (5.10)$$

Komponen-komponen pada *board AR.Drone* membutuhkan sumber listrik untuk dapat beroperasi. Melalui persamaan (2.13) dan (2.14) dan gambar 4.2 maka bisa diambil besar arus total yang digunakan oleh komponen. Persamaan 5.11 digunakan untuk mendapatkan arus total saat komponen dikonfigurasi secara parallel, sedangkan persamaan 5.12 untuk konfigurasi serial.

$$I_{total} = I_1 + I_2 + I_3 \ldots + I_n \quad (5.11)$$

$$\frac{1}{I\ total} = \frac{1}{I_1} + \frac{1}{I_2} + \frac{1}{I_3} + \cdots + \frac{1}{I_n} \quad (5.12)$$

Module Wifi :

$$\frac{1}{Wifi1} = \left(\frac{1}{PAREG\_BASE} + \frac{1}{VAT\_42} + \frac{1}{VDD33}\right) \quad (5.13)$$

$$\frac{1}{Wifi1} = \left(\frac{1}{175} + \frac{1}{175} + \frac{1}{175}\right) \quad (5.14)$$

$$Iwifi = ItotalWifi1 + AVDD18 \quad (5.15)$$

$$= 175 + 7$$

$$= 182\ mA$$

Pressure Sensor :

$$\frac{1}{Ipress} = \frac{1}{VDD} + \frac{1}{VDD\_IO} \quad (5.16)$$

$$\frac{1}{Ipress} = \frac{1}{0.005} + \frac{1}{0.005} \quad (5.17)$$

$$Ipress = 0.005\ mA$$

Accelerometer :

$$\frac{1}{Iaccelero} = \frac{1}{VDD} + \frac{1}{VDD\_IO} \quad (5.18)$$

$$\frac{1}{Iaccelero} = \frac{1}{200} + \frac{1}{200} \quad (5.19)$$

$$Iaccelero = 200\ mA$$

Compass :

$$\frac{1}{Icompass} = \frac{1}{Analog\ Power} + \frac{1}{Digital\ Interface} \quad (5.20)$$

$$\frac{1}{Icompass} = \frac{1}{350} + \frac{1}{350} \quad (5.21)$$

$$Icompass = 350mA$$

$$Itotal = Iwifi + IPower\&USB + I_{NAND} + IDDR_{SDRAM} + Ipress + Igyro + IMCU + Iaccelero + Iopamp + Icompass \quad (5.22)$$

$$Itotal = 182 + 1200 + 8 + 46 + 0.005 + 65 + 250 + 200 + 2 + 350 + 46 + 182 \quad (5.23)$$

$$= 2531\ mA$$

Setelah didapatkan nilai arus, kemudian digunakan rumus 5.23. Dan arus yang di dapat dari motor di jumlahkan dengan arus yang digunakan oleh komponen *AR.Drone*. Dan dengan merujuk pada persamaan (2.16) dan (2.17) maka dapat diketahui waktu estimasi, yang akan menunjukkan seberapa lama *quadcopter* dapat terbang.

Selanjutnya untuk dapat melakukan estimasi perpindahan, maka dilakukan percobaan untuk mengetahui besar perpindahan *quadcopter* dari titik awal selama satu menit dengan gerakan maju, mundur, ke kiri dan ke kanan. Tabel 5.4 menunjukkan hasil dari percobaan yang dilakukan sebanyak 5 kali sesuai dengan input kecepatan *quadcopter*.

Tabel 5.4 Perpindahan *Quadcopter*

Input Kecepatan	Perpindahan (m)			
	Maju	Mundur	Kanan	Kiri
0.3	21	28	27	26
0.4	23	24	24	24
0.5	32	33	27	29
0.6	50	42	55	42
0.7	47	53	49	51
	34.6	36	36.4	34.4
	Perpindahan rata-rata : 35.3 meter			

$$\overline{perpindahan\ input\ kecepatan} = \frac{\sum_{0.3}^{5} u_i}{5} \quad (5.24)$$

$$Rata - rata\ perpindahan\ akhir = \frac{maju + mundur + kiri + kanan}{4} \quad (5.25)$$

Dengan :

u_i = perpindahan yang ditempuh oleh *quadcopter* pada input kecepatan tertentu.

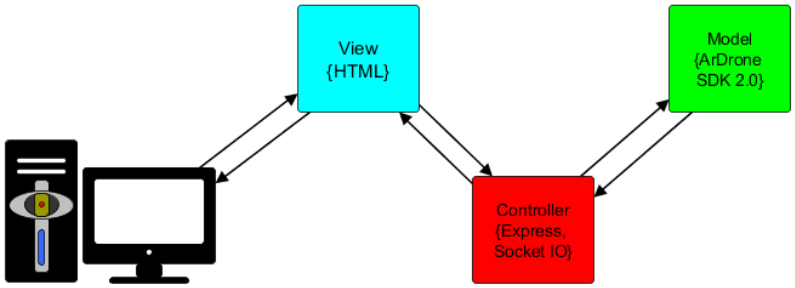
Pada tabel 5.4 ditunjukkan bahwa nilai rata-rata jarak dinyatakan kedalam satuan meter/menit, karena percobaan dilakukan selama satu menit. Dengan menggunakan persamaan 6.3 maka nilai rata-rata jarak dikonversi kedalam satuan m/s.

$$35.35\ {}^{meter}/_{menit} = \left(35.35\ \frac{meter}{\cancel{menit}}\right)\left(\frac{1}{60}\ \frac{\cancel{menit}}{detik}\right) = 0.5\ m/s \quad (5.26)$$

Satuan internasional untuk menyatakan kecepatan *linear* adalah m/s, maka dari itu nilai rata-rata perpindaha per satuan waktu yang didapat dari percobaan merupakan besaran kecepatan. Sehingga untuk dapat melakukan estimasi perpindahan maka digunakan persamaan 2.26 yang digunakan untuk mencari perpindahan benda berdasarkan kecepatan dan waktu dari titik awalnya. Sehingga didapat hasil akhir yang ditunjukkan pada persamaan

$$s = 35.35\ \times t \quad (5.27)$$

5.1.5 Perancangan Antarmuka Web



Gambar 5.11 Model View Controller

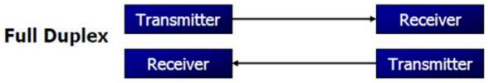
Bagan perancangan pada gambar 5.11 menjelaskan *MVC* atau *Model View Controller*, yang mana tiap komponen tersebut mempunyai kegunaan aatau peran sendiri-sendiri.

1. *Model*

Pada model digunakan Node JS yang mana terdapat paket dengan nama *AR.Drone* yang datanya bisa diakses secara langsung, terus -menerus, dan secara *realtime*.

2. *Controller*

Pada sisi controller di digunakan *Express* untuk dapat merutekan *module* serta dapat mengakses file dengan ekstensi js lainnya, selain *express.js* juga terdapat *Socket IO* yang merupakan web socket karena sistem yang dikembangkan menggunakan aplikasi berbasis realtime, dan guna dari web socket salah satunya adalah untuk dapat mengirim data tanpa adanya request terlebih dahulu sehingga sistem akan lebih efisien. Web socket mendukung fungsi *full duplex* dimana dua buah piranti saling berkomunikasi dalam waktu yang bersamaan, dimana piranti dapat bertindak sebagai transmitter dan receiver, cara komunikasi full duplex dapat ditunjukkan seperti bagan dibawah ini:



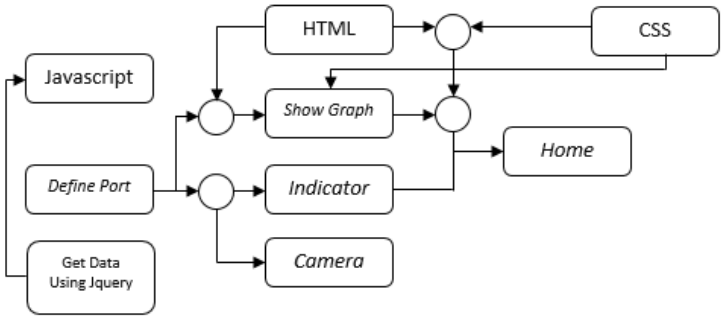
Gambar 5.12 Alur komunikasi *Full Duplex*

Sumber: Duta (2015)

Dari gambar 5.12 digambarkan alur komunikasi antara 2 buah piranti yang keduanya dapat bertindak sebagai *transmitter* dan *receiver*. Komunikasi awal dimana *transmitter* pada piranti 1 mengirim data pada *receiver* pada piranti 2, kemudian *transmitter* dari piranti 2 juga mengirim data pada *receiver* piranti satu. Namun komunikasi dapat dengan bebas tanpa menuggu *transmitter* mengirim *receiver*, contoh umum dari *full duplex* ini adalah telepon.

3. View

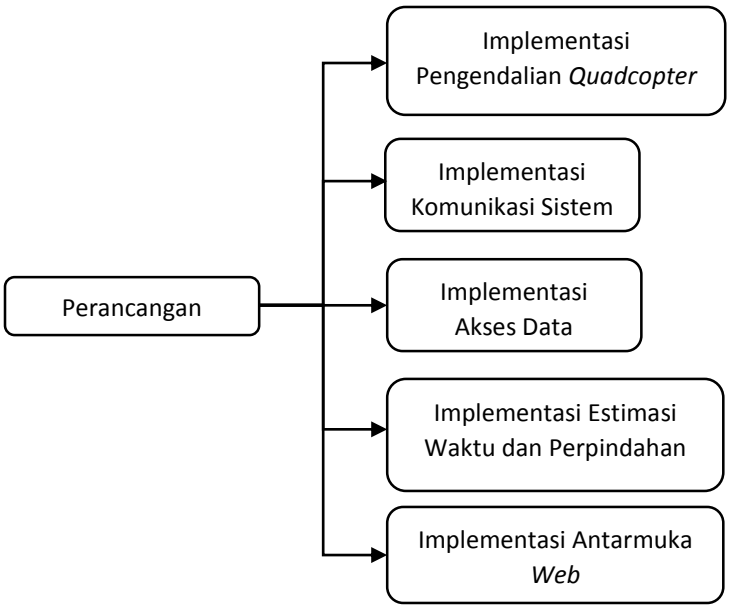
View menggunakan *HTML*, tetapi view disini tidak hanya menggunakan *html* saja melainkan juga akan menggunakan *CSS*, dan *Jquery*. Fungsi CSS adalah membuat tampilan web menjadi lebih menaarik, sedangkan fungsi Jquery adalah agar data yang telah di publish oleh socket IO dapat di tampilkan di web interface



Gambar 5.13 Model Akses Web

Selain yang telah dijelaskan pada *MVC*, pada gambar 5.8 juga menjelaskan tentang akses model. Pada sistem ini kan dibuat halaman *html*. Pada halaman home terdapat tampilan kapasitas baterai dengan model GUI, terdapat *quadcopter* indikator yang berisi angka, *navigation menu*, dan camera. Fungsi pertama dari *javascript* adalah untuk mendefinisikan port yang digunakan untuk mengakses html, yang kedua adalah menjembatani fungsi antara *Socket IO* dan *Jquery*, supaya data dari *server* dapat ditampilkan pada halaman *web*.

5.2 Implementasi Sistem



Gambar 5.14 Diagram Implementasi Sistem

Pada gambar 5.14 dijelaskan bahwa pada tahap implementasi ini dilakukan proses pembuatan kode program untuk komunikasi sistem, pembuatan kode program pengendalian *quadcopter*, kode program untuk akses data, kode program untuk estimasi waktu dan perpindahan, serta pembuatan kode program untuk antarmuka web. Dan setelah kode program dibuat, maka selanjutnya akan diintegrasikan sehingga menjadi sebuah sistem yang sesuai dengan dengan kebutuhan.

5.2.1 Implemetasi Komunikasi

Pada tahap implementasi komunikasi, tujuan yang perlu dicapai adalah terciptanya komunikasi antara *User*, *Controller*, *Web Interface* dan *AR.Drone*. Pada gambar 5.2 dijelaskan bahwa tahap awal komunikasi pengguna berusaha menyambungkan pengendalian dengan AR.Drone dengan menggunakan *wifi*. Langkah selanjutnya adalah memanggil *port* yang digunakan untuk menampilkan halaman *web*. Pengguna bisa melakukan kontrol terhadap *quadcopter*, ketika pengguna memberikan aksi kemudian *AR.Drone* memberika reaksi. *Quadcopter* yang terbang atau melakukan operasi lainnya akan mengubah indikator yang bisa

diamati oleh pengguna sehingga pengguna mengetahui kapan batas untuk dapat melakukan pengisian ulang selanjutnya.

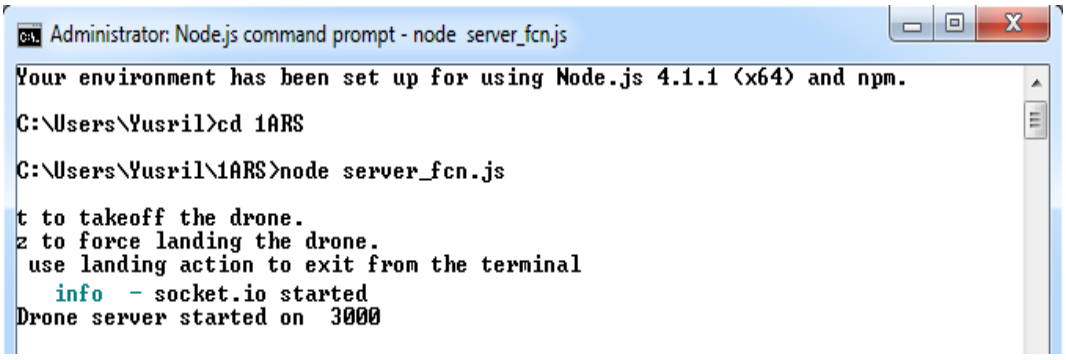
Komunikasi dapat terbentuk dimulai dari komputer/laptop yang terhubung dengan *quadcopter* menggunakan protokol *UDP*. Untuk dapat berkomunikasi dengan *quadcopter* pertama perlu menyambungkan *Wi-Fi* dimana *IP* yang diset adalah 192.168.1.1 untuk *quadcopter* dan *IP* untuk klien adalah 192.168.1.2. Setelah tersambung perlu melakukan konfigurasi. Konfigurasi dilakukan dengan menuliskan kode program pada tabel 5.5.

Tabel 5.5 kode Program Konfigurasi Awal

```
1  var ardrone = require("ar-drone");
2  var client = ardrone.createClient();
3
4  client.config = ("general:navdata_demo", FALSE);
5  client.on('navdata', function(navdata){
6  });
```

Tabel 5.6 Pembahasan Kode Program Konfigurasi Awal

1.	Deklarasi *variable* dengan nama ardrone, yang memanggil paket SDK Node Ar.Drone 2.0.
2.	Deklarasi *variable* dengan nama *client*, yang berisi module client Node Ar.Drone 2.0.
4.	Set konfigurasi agar masuk dalam *mode bootstrap*, sehingga dapat mengakses *object* selain yang ada pada *object demo.*
5-6	Proses untuk memanggil object yang digunakan untuk memantau navdata.



Gambar 5.15 Proses Koneksi Antara AR.Drone dan Komputer

Gambar 5.16 merupakan cara melakukan koneksi antara AR.Drone dengan Komputer. Langkah pertama yang dilakukan adalah masuk kedalam *directory* program utama. Setalah itu panggil dengan syntax "`node "nama_program.js""`, apabila koneksi berhasil maka terdapat status yang menujukkan bahwa program dapat dipanggil pada port yang telah di set pada program utama, pada web browser dengan mengetikkan "localhost:PORT" seperti pada gambar 5.17.

Gambar 5.16 Koneksi pada *web*

5.2.2 Implementasi Pengendalian *Quadcopter*

Tabel 5.7 Kode Program Pengendalian *Quadcopter*

```
1  keypress(process.stdin);
2  process.stdin.on('keypress', function(ch, key) {
3      if (key && key.ctrl && key.name == 'c') {
4          process.exit(0);
5      }
6        else if(key && key.name == 't') {
7              client.takeoff();
8              console.log('Force takeoff the drone.');
9        }
10
11       else if(key && key.name == 'z') {
12             client.stop();
13             client.land();
14             console.log('Force landing the drone.');
15       }
16
17       else if(key  && key.name == 'm') {
18             client.stop();
19             console.log('Drone is now hovering');
20     }
21
22       else if(key  && key.name == 'u') {
23             client.up(speed);
24             console.log('going  up  with  '+  speed  +  '
25 speed');
26     }
27
28       else if(key  && key.name == 'j') {
29             client.down(speed);
30             console.log('going  down  with  '+  speed  +  '
31 speed');
32     }
33
34       else if(key  && key.name == 'w') {
35             client.front(speed);
36             console.log('going forward with '+ speed + '
37 speed');
38     }
39
40       else if(key  && key.name == 'a') {
41             client.left(speed);
42             console.log('going  to  left  with  '+  speed  +
43 ' speed');
```

```
    }

     else if(key  && key.name == 's') {
          client.back(speed);
          console.log('going backward with '+ speed +
' speed');
    }

     else if(key  && key.name == 'd') {
          client.right(speed);
          console.log('going right with '+ speed + '
speed');
    }

     else if(key  && key.name == 'b') {
          client.clockwise(speed);
          console.log('turn  right  '+  speed  +  '
speed');
    }

     else if(key  && key.name == 'n') {
          client.counterClockwise(speed);
          console.log('turn left '+ speed + ' speed');

    }

     else if(key && key.name == 'p') {
          if (speed >= 0.9){
               speed=0.3;
          } else {
          speed= (speed+adder);
          }
          client.stop();
          console.log('menambah kecepatan');
    }

     else if(key && key.name == 'o') {
          if (speed <= 0.3){
          speed=0.3;
          } else {
          speed= (speed-adder);
          }
          client.stop();
          console.log('mengurangi kecepatan');
    }
});

process.stdin.setRawMode(true);
process.stdin.resume();
```

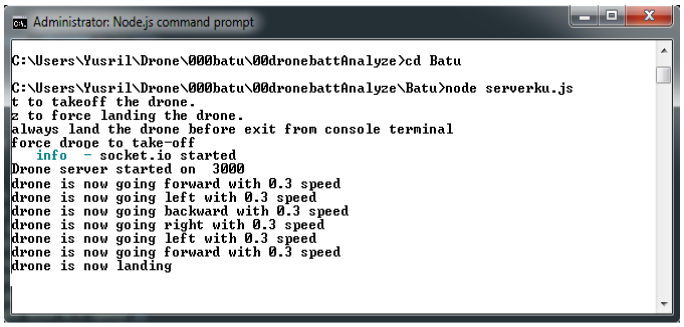
Tabel 5.8 Kode Program Pengendalian *Quadcopter*

1.	Deklarasi *keypress*
2.	Membuta fungsi, yang berisi *parameter input keypress.*
3-5	*Input keypress*, kondisi jika tombol *keyboard* C ditekan, maka keluar dari sistem.
6-9	*Input keypress,* kondisi jika tombol *keyboard* T ditekan, maka quadcopter akan *takeoff* atau lepas landas.
11-15	*Input keypress*, kondisi jika tombol keyboard Z ditekan, maka quadcopter akan mendarat/*landing*.
17-20	*Input keypress*, kondisi jika tombol *keyboard* M ditekan, maka *quadcopter* akan diam atau hover.
22-25	*Input keypress*, kondisi jika tombol *keyboard* U ditekan, maka *quadcopter* akan naik ke atas.
28-32	*Input keypress*, kondisi jika tombol *keyboard* J ditekan, maka *quadcopter* akan turun kebawah.
34-38	*Input keypress*, kondisi jika tombol *keyboard* W ditekan, maka *quadcopter* akan maju kedepan.
40-44	*Input keypress*, kondisi jika tombol *keyboard* A ditekan, maka *quadcopter* akan bergerak kekiri.
46-50	*Input keypress*, kondisi jika tombol *keyboard* S ditekan, maka *quadcopter* akan bergerak ke belakang.
52-56	*Input keypress*, kondisi jika tombol *keyboard* D ditekan, maka *quadcopter* akan bergerak ke kanan.
58-62	*Input keypress*, kondisi jika tombol *keyboard* B ditekan, maka *quadcopter* akan bergerak *clockwise*(Searah jarum jam)
64-68	*Input keypress*, kondisi jika tombol *keyboard* N ditekan, maka *quadcopter* akan bergerak *counterclockwise*
70-78	*Input keypress*, kondisi jika tombol *keyboard* P ditekan, maka *quadcopter* akan menambah input kecepatan sebesar 0.1 dalam sekali tekan
80-89	*Input keypress*, kondisi jika tombol *keyboard* P ditekan, maka *quadcopter* akan menambah input kecepatan sebesar 0.1 dalam sekali tekan.
91	Baris kode yang digunakan agar dapat menginputkan tombol keyboard Baris kode yang digunakan agar dapat meng-*interrupt input keypress*

Tabel 5.7 kode program pengendalian *quadcopter* adalah program yang digunakan untuk dapat menerbangkan *quadcopter* dengan operasi maju, mundur, keatas, kebawah, ke kanan, ke kiri, berputar searah jarum jam, dan berputar berlawanan arah jarum jam. Dalam sistem ini juga ditambah fitur untuk dapat mengurangi dan menambah kecepatan melalui inputan dari *keyboard*. Karena penggunaannya yang tidak memerlukan piranti lain, dan penggunaannya sama dengan *controller game* pada umumnya sehingga memudahkan user untuk dapat

menerbangkannya menjadi pertimbangan mengapa memilih *keyboard* sebagai kontroler.

Kode program pada tabel 5.7 mengenai pengendalian *quadcopter* ditulis pada program utama. Dan karena masuk pada server *Node JS* maka perlu dilakukan proses pemanggilan kode program. Proses pengendailian itu dilaksanakan seperti pada gambar 5.4



Gambar 5.17 Implementasi Pengendalian *Quadcopter*

Dari gambar 5.18 dapat diketahui bahwa ketika *quadcopter* terbang maka *console terminal* akan memberitahukan status gerakan yang sedang dilakukan *quadcopter*. Ketika *server* dipanggil *module keypress* akan secara otomatis aktif sehingga langkah pertama yang harus dilakukan adalah *take-off.* Jika sudah melakukan *take-off* pengguna dapat melakukan operasi terbang, dan dapat menambah kecepatan secara simultan.

5.2.3 Implementasi Akses Data

Pada tahap perancangan pengiriman data, terdapat proses yang mengambil *state bit mask*. *State bit mask* dapat diketahui pada kode program dalam tabel 5.9.

Tabel 5.9 Kode Program *constan.option*

```
1   constants.options = {
2     DEMO              : 0,
3     TIME              : 1,
4     RAW_MEASURES      : 2,
5     PHYS_MEASURES     : 3,
6     GYROS_OFFSETS     : 4,
7     EULER_ANGLES      : 5,
8     REFERENCES        : 6,
9     TRIMS             : 7,
10    RC_REFERENCES     : 8,
11    PWM               : 9,
12    ALTITUDE          : 10,
13    VISION_RAW        : 11,
14    VISION_OF         : 12,
15    VISION            : 13,
16    VISION_PERF       : 14,
17    TRACKERS_SEND     : 15,
18    VISION_DETECT     : 16,
```

```
19    WATCHDOG          : 17,
20    ADC_DATA_FRAME    : 18,
21    VIDEO_STREAM      : 19,
22    GAMES             : 20,
23    PRESSURE_RAW      : 21,
24    MAGNETO           : 22,
25    WIND_SPEED        : 23,
26    KALMAN_PRESSURE   : 24,
27    HDVIDEO_STREAM    : 25,
28    WIFI              : 26,
29    ZIMMU_3000        : 27,
30    CKS               : 65535
31  };
```

Tabel 5.10 Pembahasan Kode Program *Constant.option*

1.	*Object constant.option*
2.	*Object Demo*
3.	*Object Time*
4	*Object Raw_measures*
5	*Object Phys_Measures*
6	*Object Gyros_offsets*
7	*Object Euler_angels*
8	*Object Trims*
9	*Object RC_refrences*
10	*Object PWM*
11	*Object Altitude*
12	*Object Vision Raw*
13	*Object Vision_Of*
14	*Object object Vision*
15	*Object Vision Pref*
16	*Object Trackers_Send*
17	*Object Vision_Detect*
18	*Object Watchdog*
19	*Object Adc_Data_Frame*
20	*Object Video_Stream*
21	*Object Games*
22	*Object Pressure_Raw*
23	*Object Magneto*
24	*Object Wind_Speed*
25	*Object Kalman_Pressure*
26	*Object Hdvideo_Stream*
27	*Object Wifi*
28	*Object Zimmu_3000*

Constant.option yang digunakan terdapat pada *module constant*. Untuk dapat mengaksesnya kita perlu menggunakan AT *Command*, dan yang digunakan adalah `AT*CONFIG` dan `AT*CONFIG_IDS`. Konfigurasi *navdata* dalam sistem ini diintegrasikan secara langsung dengan *websocket* karena tujuannya adalah mengirimkan *navdata* kedalam halaman *web*, jika tanpa menggunakan *web socket* maka *navdata* hanya akan bisa diakses dalam *CLI* (*Command Line Interface*). Pada program 5.4 merupakan kode program yang digunakan untuk dapat mengakses *navdata* yang ditulis kedalam bahasa pemrograman *Javascript*.

Tabel 5.11 Kode Program Akses *Node JS Navdata*

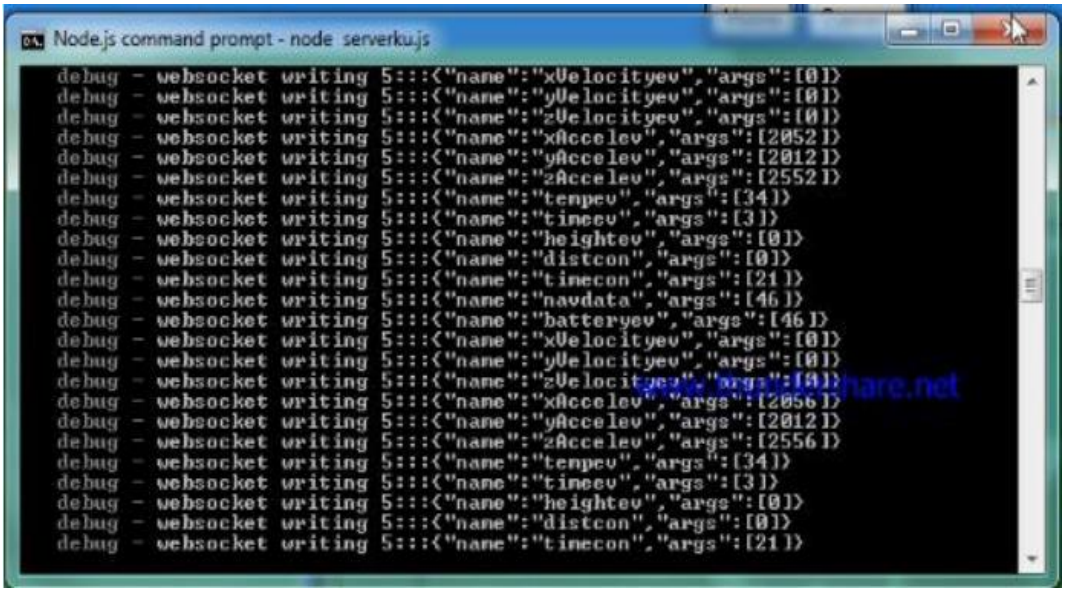
```
1   client.config('general:navdata_demo', 'FALSE');
2
3   var port = process.env.PORT || 3000;
4   app.set('port', port);
5   app.use(express.static(path.normalize
    (__dirname + '/')));
6   var http = require('http');
7   var server = http.createServer(app);
8
9   var sio = io.listen(server);
10  sio.set('destroy upgrade', false);
11  sio.sockets.on('connection', function (socket) {
12  console.log('New Socket.IO connection');
13  client.on('navdata', function(data) {
14
15  var nowMs = new Date().getTime();
16  if (nowMs - lastNavDataMs > 1000) {
17  lastNavDataMs = nowMs;
18  if(data.demo){
19  var baterai = data.demo.batteryPercentage;
20  socket.emit('navdata', baterai);
21  socket.emit('battery_value',{name:battery,value:
     baterai});
22  var angular_speedX =(Math.round(data.demo.
23  velocity.x));
24  socket.emit('xVelocity_value', angular_speedX);
25  var angular_speedY = (Math.round(data.demo.
    velocity.y));
26  socket.emit('yVelocity_value', angular_speedY);
27  var angular_speedZ = (Math.round(data.demo.
    velocity.z));
28  socket.emit('zVelocity_value', angular_speedZ);
29
30  var pitchOrientation = data.demo.frontBackDegrees;
31  socket.emit('pitchValue', pitchOrientation);
32  var rollOrientation = data.demo.leftRightDegrees;
33  socket.emit('rollValue', rollOrientation);
34  var yawOrientation = data.demo.clockwiseDegrees;
35  socket.emit('yawValue', yawOrientation);
36
37  var ketinggian = data.demo.altitudeMeters;
```

```
38  socket.emit('height_value', {name:drone_height,
    value: ketinggian});
39  }
40  if(data.pressureRaw){
41  var suhu = (Math.round(data.pressureRaw.
    temperature/10));
42  socket.emit('temperature_value', {name: temperature,
    value: suhu});
43  }
44  if(data.time){
45  var waktu_aktif = (Math.round((data.time/1000)/60));
46  socket.emit('time_active', {name: time_activated,
    value: waktu_aktif});
47  }
```

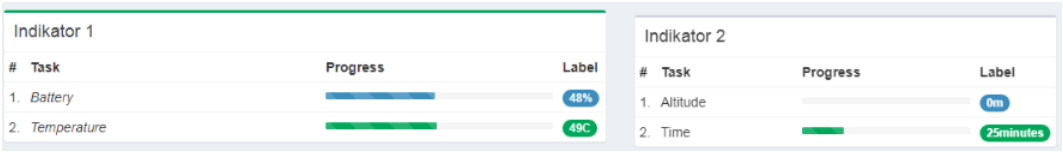
Tabel 5.12 Pembahasan Kode Program Akses Node JS *Navdata*

1.	Set konfigurasi keluar dari *mode demo*.
3.	Deklarasi *variable* port pada PORT 3000
4.	Inisialiasi *port*
5-6	Konfigurasi penggunaan *express*, untuk *routing*.
7	Deklarasi *variable http* yang memanggil *module http*
8	Deklarasi *variable* server, dan memanggil *port*.
10	Deklarasi *variable web socket*.
11	*Turn off* mode *destroy*
12	*Turn on* socket IO, dan membuat fungsi dengan nama *parameter socket*
15	*Turn on* client, dan set fungsi, dengan *parameter* bernama *data*.
17-19	Set *Milis*, dengan nilai 1 detik.
21-29	*Publish socket* yang bernilai kapasitas baterai dinyatakan dalam %
31-34	*Publish socket* yang bernilai kecepatan *linear* x dalam satuan mm/s^2
36-39	*Publish socket* yang bernilai kecepatan *linear* y dalam satuan mm/s
41-44	*Publish socket* yang bernilai kecepatan *linear* z dalam satuan mm/s
46-49	*Publish socket* yang bernilai percepatan *linear* x dalam satuan mm/s^2
51-54	*Publish socket* yang bernilai percepatan *linear* y dalam satuan mm/s^2
55-58	*Publish socket* yang bernilai percepatan *linear* z dalam satuan mm/s^2
59-63	*Publish socket* yang bernilai temperatur dalam satuan derajat *Celcius*
65-69	*Publish socket* yang bernilai waktu dalam satuan menit
70-73	*Publish socket* yang bernilai ketinggian dalam satuan meter

Sama seperti konfigurasi komunikasi dan pengendalian. Proses pengiriman *navdata* juga terdapat pada program utama. Pengimplementasian data akan dikirim pada antarmuka *web* dan pada *console terminal*. Dan dapat ditunjukkan pada gambar 5.15 dan gambar 5.16.



Gambar 5.18 Tampilan *Console Terminal*



Gambar 5.19 Tampilan Antarmuka *web*

5.2.4 Implementasi Estimasi Perpindahan dan waktu

Untuk bisa mengetahui estimasi waktu dan perpindahan, perlu mengakses beberapa data dari *navdata* itu sendiri. Data yang diambil adalah kapasitas baterai dan besar arus yang mengalir saat *quadcopter* beroperasi. Untuk itu sebelum bisa mengolah data, terlebih dahulu data tersebut harus bisa diakses secara bersamaan. Berikut ini adalah kode program implementasi estimasi waktu dan perpindahan.

Tabel 5.13 Kode Program Estimasi Perpindahan dan Waktu

```
1   if(data.pwm && data.demo)
2   var pwmar = data.pwm.motors;
3   var motor1 = pwmar[0];
4   var motor2 = pwmar[1];
5   var motor3 = pwmar[2];
6   var motor4 = pwmar[3];
7
8   function arusmotor(motor_pwm){
9   var dutyCycle = motor_pwm*setdutyCycle/setpwm;
10  var dutyCycle_percentage    = dutyCycle/100;
11  var tegangan = dutyCycle_percentage *11.1;
12  var arus = tegangan/8.5;
13  return arus;
14  }
15  var arus_motor1 = arusmotor(motor1);
16  var arus_motor2 = arusmotor(motor2);
17  var arus_motor3 = arusmotor(motor3);
```

```
var arus_motor4 = arusmotor(motor4);
var arus_total = (((arus_motor1 + arus_motor2 +
arus_motor3 + arus_motor4)*1000).toFixed(2));
var batteryValue = data.demo.batteryPercentage;
var findcapacity = ((batteryValue * setcapacity)
/setbatteryPercentage);
var time_estimating=(findcapacity/(arus_
total+2238)*60); var vetotal = 35.5;
var distance = (Math.round(vetotal*(time)));
socket.emit('distance_estimation', distance);
socket.emit('time_estimation', time_estimating);
}
```

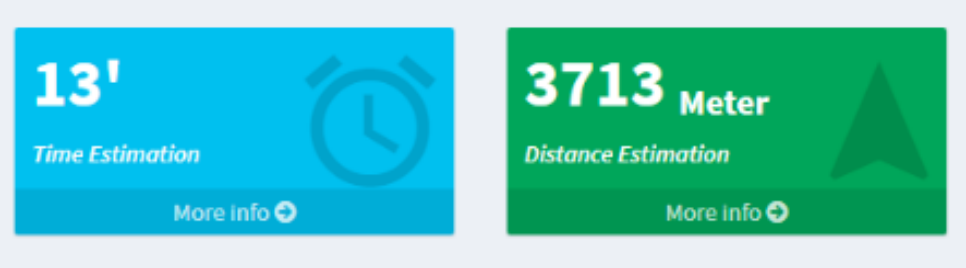
Tabel 5.14 Pembahasan Kode Program Estimasi Perpindahan dan Waktu

1	Kondisi jika *object* `data.pwm` dan `data.demo` berhasil diakses
2	Deklarasi *variable* pwmar yang bernilai *parameter data.pwm* dari *object* .
3	Deklarasi variable dengan nama motor1 yang bernilai *index* pertama dari *parameter data.pwm*
4	Deklarasi variable dengan nama motor2 yang bernilai *index* kedua dari *parameter data.pwm*
5	Deklarasi variable dengan nama motor3 yang bernilai *index* ketiga dari *parameter data.pwm*
6	Deklarasi variable dengan nama motor4 yang bernilai *index* keempat dari *parameter data.pwm*
8	Deklarasi variable, yang melakukan proses perhitungan *duty cycle* pada motor1.
9	Deklarasi variable, yang melakukan proses perhitungan *duty cycle* pada motor1 dalam satuan persen.
10	Deklarasi variable tegangan, yang melakukan proses perhitungan dari duty cycle dan persentasinya pada motor 1.
11	Deklarasi variable Arus, yang melakukan proses perhitungan dari tegangan dan resistansi pada motor1.
13	Deklarasi variable, yang melakukan proses perhitungan *duty cycle* pada motor2.
14	Deklarasi variable, yang melakukan proses perhitungan *duty cycle* pada motor2 dalam satuan persen.
15	Deklarasi variable tegangan, yang melakukan proses perhitungan dari duty cycle dan persentasinya pada motor2.
16	Deklarasi variable Arus, yang melakukan proses perhitungan dari tegangan dan resistansi pada motor2.
18	Deklarasi variable, yang melakukan proses perhitungan *duty cycle* pada motor3.
19	Deklarasi variable, yang melakukan proses perhitungan *duty cycle* pada motor3 dalam satuan persen.
20	Deklarasi variable tegangan, yang melakukan proses perhitungan dari duty cycle dan persentasinya pada motor3.

23	Deklarasi variable Arus, yang melakukan proses perhitungan dari tegangan dan resistansi pada motor3.
24	Deklarasi variable, yang melakukan proses perhitungan *duty cycle* pada motor4.
25	Deklarasi variable, yang melakukan proses perhitungan *duty cycle* pada motor4 dalam satuan persen.
26	Deklarasi variable tegangan, yang melakukan proses perhitungan dari duty cycle dan persentasinya pada motor4.
28-29	Deklarasi variable Arus, yang melakukan proses perhitungan dari tegangan dan resistansi pada motor4.
	Nilai penjumlahan dari ke-4 motor.
	Deklarasi *variable* yang mengakses nilai kapasitas baterai.
30	Mendapatkan nilai dari estimasi waktu.
32-34	Mendapatkan nilai estimasi perpindahan.
36-43	

Pengguna arus paling besar dan paling berpengaruh pada *quadcopter* adalah *motor brushless* arus searah. Putaran 4 motor arus searah pada *quadcopter* diatur menggunakan PWM, melalui perancangan dan analisis maka bisa didapatakan nilai arus dari ke-4 motor serta komponen utama yang memerlukan arus.

Data *PWM* dari ke-4 motor disimpan dalam *data* berupa *array*, yang menyatakan ke-4 motor tersebut [motor1, motor2, motor3, motor4]. *Index array* dimulai dari *index* ke-0 sehingga untuk mengakses motor 1 hingga 4, digunakan penanggilan *index array* mulai 0 -3. Setelah diambil nilai arus dari ke-4 motor maka kemudian diolah. Hasil dari implementasi perhitungan perpindahan dan waktu estimasi ditunjukkan pada gambar 5.18 dimana kotak dibagian atas akan menunjukkan estimasi waktu, sedangkan kotak yang berada dibawah menujukkan estimasi perpindahan. Dimana estimasi waktu sangat berhubungan dengan sisa kapasitas baterai yang tersisa, dimana batas kapasitas paling rendah dari baterai yang dapat digunakan adalah 15%, sedangkan perpindahan estimasi didapat dari percobaan yang ditunjukkan pada tabel 6.3 yang dikalikan dengan nilai estimasi waktu yang telah didapatkan.



Gambar 5.20 Impelementas Estimasi Perpindahan dan Waktu

5.2.5 Implementasi Antarmuka Web

Masuk kedalam bagian akhir proses implementasi. Interface web berguna untuk dapat menjembatani interaksi antara pengguna dan *quadcopter*. Tidak seperti *CLI* yang hanya bisa menampilkan tampilan data berupa string, tampilan *web* dapat lebih variatif, dan informatif karena dalam sistem ini juga menggunakan tampilan berupa *Graphical User Interface*. Untuk mengiplementasikan tampilan web digunakan *HTML, CSS*, dan *Javascript*, dan *jquery* yang masih satu kelompok dengan *javascript*.

Tabel 5.15 Kode Program HTML

```
1   <!DOCTYPE html>
2   <html>
3   <body class="hold-transition skin-blue sidebar-mini">
4   <div class="wrapper">
5   <h3>53<span id="timefly"></span>'</h3>
6   <b><i>Time Estimation</i></b>
7   <h3>53 <span id="distancefly"></span> <sub style="font
    -size: 20px">Meter</sub></h3>
8   <b><i>Distance Estimation</i></b>
9   </div>
10  <div>x : <span id="xVelocity"></span>
11  <span>mm/s</span></div>
12  <div>y : <span id="yVelocity"></span>
13  <span>mm/s</span></div>
14  <div>z : <span id="zVelocity"></span>
15  <span>mm/s</span></div>
16  <b><i>Linear Velocity</i></b>
17  </div>
18  <div>&#x3B8 : <span id="pitch"></span></div>
19  <div>&#x3C6 : <span id="roll"></span></div>
20  <div>&#x3A8 : <span id="yaw"></span></div>
21  <b><i>Orientation</i></b>
22  </div><div class="epoch text-center" id="lineChart"
23  style="height: 250px; width: 600px; position:
24  relative;">
25  <div class="progress-bar progress-bar-primary"
    id="battery-indicator" style=" width;"></div>
26  </div>
27  <td><span class="badge bg-light-blue" id="battery-
28  value">%</span></td>
29  <td>2.</td>
30  <td><i>Temperature</i></td>
31  <div class="progress-bar progress-bar-success"
    id="temperature-indicator" style="width;"></div>
32  </div>
33  <td><span class="badge bg-green" id="temperature-
    value">%</span></td>
34  <div id="droneStream" style="width: 200px; height:
    208px"></div>
35  <script>newNodecopterStream(document.getElementById("d
    roneStream"), {port: 3001});
```

```
</script>
<td>1.</td>
<td>Altitude</td>
<div class="progress-bar progress-bar-primary"
id="height-indicator" style="width;"></div>
</div>
<td>2.</td>
<td>Time</td>
<div class="progress progress-xs progress-striped
active">
<div class="progress-bar progress-bar-success"
id="time_active-indicator" style="width;"></div>
</div>
<td><span class="badge bg-green" id="time_active-
value">%</span></td>
```

Tabel 5.16 Pembahasan Kode Program Estimasi Perpindahan dan Waktu

3-6	Kode yang digunakan untuk memanggil nilai dari perhitungan estimasi waktu
7-8	Kode yang digunakan untuk memanggil nilai estimasi perpindahan
10-16	Kode untuk menampilkan nilai kecepatan *linear x, y, z*
17-21	Kode menampilkan nilai orientasi/ posisi sudut *pitch*, *roll*, dan *yaw*
22-24	Kode untuk menampilkan nilai kapasitas baterai dalam bentuk grafik.
25-28	Kode untuk menampilkan nilai kapasitas baterai dalam bentuk progress bar.
29-33	Kode untuk menampilkan nilai suhu mesin *quadcopter dengan progress bar*.
34-36	Kode untuk menampilkan kamera
37-40	Kode yang digunakan untuk menampilkan nilai ketinggian dalam *progress bar*.
42-48	Kode yang digunakan untuk menampilkan waktu aktif pada *quadcopter*

Untuk dapat mengambil nilai *navdata* dari *host* digunakan fungsi *Jquery* yang mana penggunaannya terdapat pada *javascript*. Karena untuk dapat mengakses *navdata* dari *host* menuju ke *host* menggunakan *web socket* yang berfungsi sebagai *publisher*, dan *jquery* bertindak sebagai *subscriber*. Berikut adalah kode program penggunaan *jquery*.

Tabel 5.17 Kode Program *Jquery*

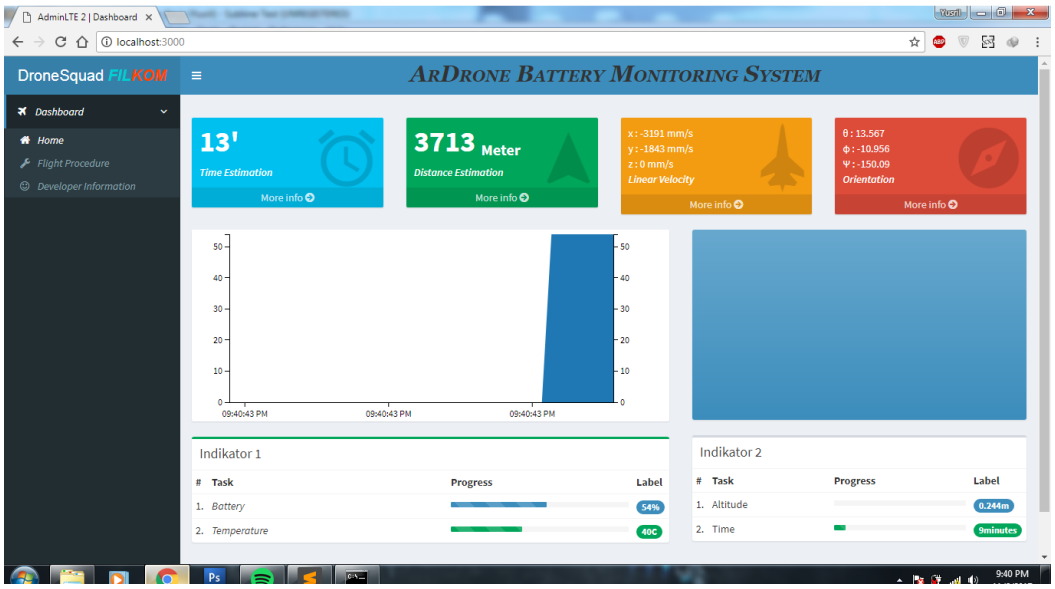
```
socket.on('batteryev', function (data){
if(data.name=="battery"){
$("#battery-indicator").css('width',data.value+'%');
$("#battery-value").html(data.value+'%');
}
});
socket.on('xVelocity_value', function (data) {
$('#xVelocity').text(JSON.stringify(data));
});
socket.on('yVelocity_value', function (data) {
$('#yVelocity').text(JSON.stringify(data));
});
socket.on('pitchValue', function (data) {
$('#pitch').text(JSON.stringify(data))
```

```
});
socket.on('rollValue', function (data) {
$('#roll').text(JSON.stringify(data));
});
socket.on('yawValue', function (data) {
$('#yaw').text(JSON.stringify(data));
});
$("#temperature-value").html(data.value+'%');}});
socket.on('time_active', function (data) {
if(data.name=="time_activated"){
$("#time_active-
indicator").css('width',data.value+'%');
$("#time_active-value").html(data.value+'%');
}
});
socket.on('height_value', function (data) {
if(data.name=="drone_height"){
$("#height-indicator").css('width',data.value+'%');
$("#height-value").html(data.value+'%');
}
});
socket.on('time_estimation', function (data) {
$('#timefly').text(JSON.stringify(data));
});
socket.on('distance_estimation', function (data) {
$('#distancefly').text(JSON.stringify(data));
});
<script>
var chart = null;
var data = [{
label: "Battery",
values: []}
];
for (var i = 0; i < 99; i++) {data[0].values.push
({time: parseInt(new Date().getTime() / 1000), y:
0});}
var socket = io.connect();
socket.on('navdata', function (d) {
if (chart && d) {
chart.push([
{time: parseInt(new Date().getTime() / 1000), y: d}]);
}
});
socket.on('batteryev', function (data) {
if(data.name=="battery"){$("#battery-
indicator").css('width',data.value+'%');
$("#battery-value").html(data.value+'%')
```

Tabel 5.18 Pembahasan Kode Program 5.9 Kode *Jquery*

1	Deklarasi socket IO.
2-6	*Subscribe* nilai kapasitas baterai

7-9	*Subscribe* nilai kecepatan angular x
10-12	*Subscribe* nilai kecepatan angular y
13-15	*Subscribe* nilai kecepatan angular z
16-18	*Subscribe* nilai orientasi pada sumbu x (*pitch*)
19-21	*Subscribe* nilai orientasi pada sumbu y (*roll*)
22-24	*Subscribe* nilai orientasi pada sumbu z (*yaw*)
25-27	*Subscribe* nilai suhu
28-34	*Subscribe* nilai waktu aktif
35-40	*Subscribe* nilai ketinggian
41-43	*Subscribe* nilai estimasi waktu
44-46	*Subscribe* nilai estimasi waktu
48	Deklarasi *variable chart*
49-62	Deklarasi variable data, dan pembuatan label
53-62	Proses perulangan, dan pembuatan interval harga x dan y pada grafik.
63-65	Subscribe nilai kapasitas baterai dalam bentuk grafik



Gambar 5.21 Tampilan Antarmuka *web*

Hasil dari kode program yang telah di integrasi ditunjukkan seperti gambar 5.22. Terdapat gfrafik yang menampilkan kinerja baterai sehingga lebih variatif dan memudahkan pemantauan. Terdapat beberapa indikator yang menunjukkan status *quadcopter*, nilai estimasi waktu dan perpindahan yang terletak pada kotak paling atas.

BAB 6 PENGUJIAN DAN ANALISIS HASIL

6.1 Pengujian Ketepatan Gerakan *Quadcopter*

6.1.1 Tujuan Pengujian

Pengujian ini bertujuan untuk melihat ketepatan gerakan *quadcopter* dengan inputan berupa *keyboard* laptop. Gerakan yang dilakukan adalah maju, mundur, ke kiri, ke kanan, naik dan turun. Dengan melakukan pengamatan apakah *key* yang digunakan untuk memberikan perintah gerakan pada *quadcopter* akan sesuai dengan yang diperintahkan.

6.1.2 Pelaksanaan Pengujian Ketepatan Gerakan *Quadcopter*

Pelaksanaan pengujian dilakukan dengan cara menyambungkan *quadcopter* dengan laptop. Apabila laptop dan *quadcopter* telah terhubung maka selanjutnya dilakukan pemanggilan program utama pada *console terminal*. Untuk dapat melakukan pengendalian *console terminal* dibiarkan terbuka

6.1.3 Prosedur Pengujian Ketepatan Gerakan *Quadcopter*

1. Menyalakan *quadcopter* hingga terdengar beep 4 kali dan *LED* berwarna hijau

2. Memasang *hull*(pelindung)

3. Menyambungkan *quadcopter* dan laptop agar dapat berkomunikasi

4. Memasuki direktori *file* dengan ekstensi .js

5. Mengetikkan *syntax* “node “nama_file”.js”

6. Memastikan *console terminal* memberikan feedback, berupa panduan untuk lepas landas dan mendarat menggunakan keyboard.

7. Memastikan baterai yang digunakan cukup.

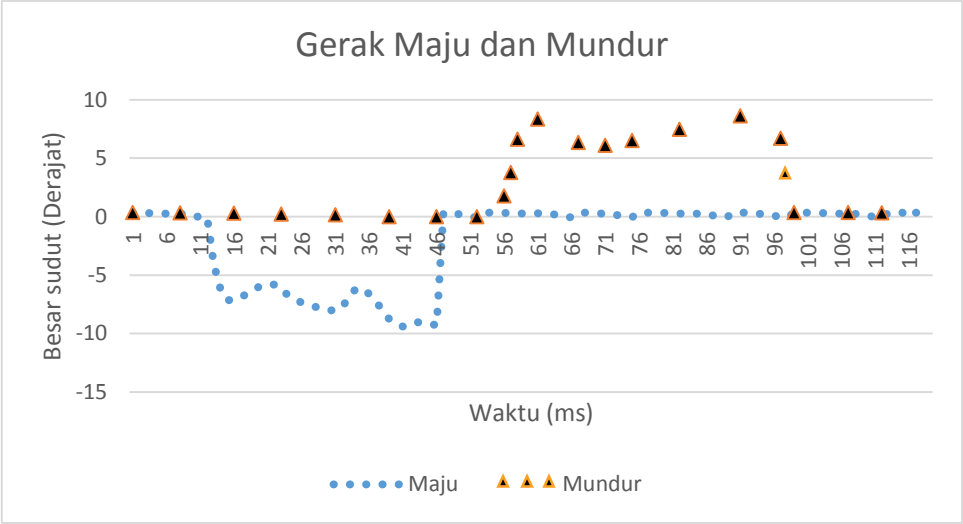
8. Melakukan perintah untuk lepas landas.

9. Melakukan operasi terbang seperti maju, mundur, ke kirim dan ke kanan dengan menggunakan *keyboard*.

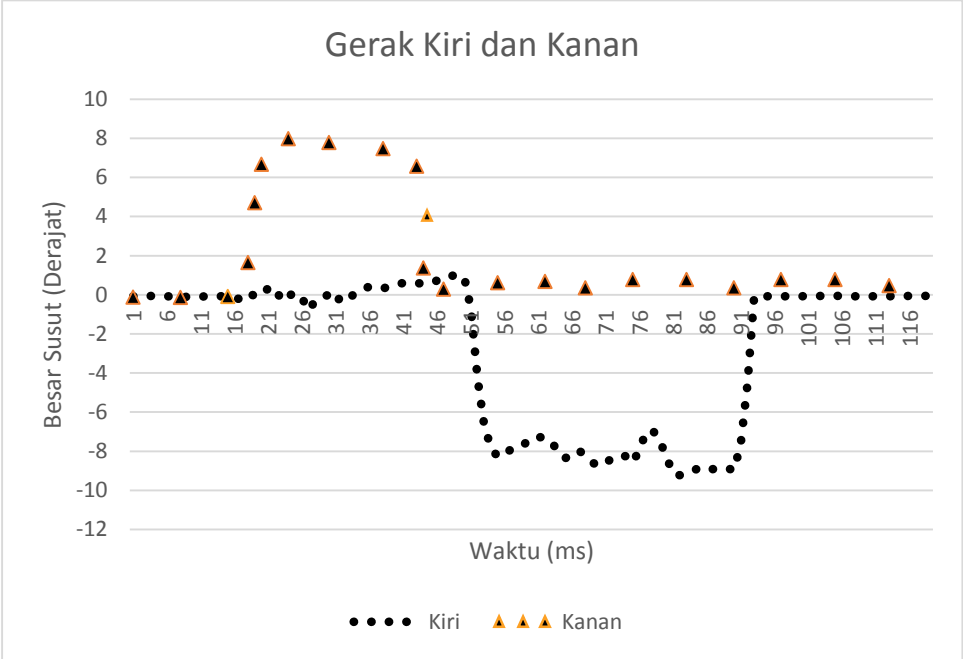
10. Mengakhiri dengan memberikan perintah mendarat.

6.1.4 Hasil Pengujian Ketepatan Gerakan *Quadcopter*

Hasil dari pengujian yang telah dilakukan menujukkan bahwa *quadcopter* akan melakukan gerakan sesuai dengan apa yang pengguna *input*-kan dengan menggunakan keyboard. Pengujian ini dilakukan dengan melakukan gerakan maju, mundur, ke kiri dan ke kanan sebanyak 10 kali. Gerakan tersebut ditunjukkan dengan gambar 6.1 dan 6.2 yang direprensentasikan kedalam bentuk grafik. Gerakan maju ditunjukkan dengan sudut bernilai negatif, sedangkan gerakan mundur ditunjukkan dengan sudut bernilai positif. Gerakan ke kiri ditunjukkan dengan sudut bernilai negatif, dan gerakan ke kanan ditunjukkan dengan sudut bernilai positif.



Gambar 6.1 Gerak Maju dan Mundur



Gambar 6.2 Gerakan Kekiri dan Kekanan

Gambar 6.1 dan 6.2 merupakan grafik dari nilai yang merepresentasikan gerakan dari quadcopter. Dengan sumbu x yang menunjukkan waktu dalam satuan ms (*millisecond*), dan sumbu y yang menujukkan besar sudut dalam satuan derajat. Gerakan maju dan ke kiri ditunjukkan dengan sumbu y yang bernilai negatif, sedangkan mundur dan ke kanan ditunjukkan dengan sumbu y yang bernilai positif. Bentuk segitiga menunjukkan grafik dengan gerakan mundur dan ke kanan, sedangakan titik dengan warna hitam menunjukkan gerakan maju dan ke kiri.

6.1.5 Analisis Hasil Pengujian Ketepatan Gerakan *Quadcopter*

Simbol *checked* pada tabel 6.1 menunjukkan bahwa pengujian ketepatan gerakan *quadcopter* yang dilakukan dengan menggunakan *keyboard* laptop menunjukkan hasil yang baik. Dari ke-empat gerakan yang diuji sebanyak 10 kali diketahui bahwa presentase kesalahan ketepatan gerakan pada *quadcopter* adalah 0%. Nilai *error* tersebut didapat dari penghitungan dari persamaan 6.1.

Tabel 6.1 Pengujian Ketepatan Gerakan *Quadcopter*

Gerakan	Percobaan ke-									
	1	2	3	4	5	6	7	8	9	10
Maju	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Mundur	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Ke Kiri	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Ke Kanan	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

$$Presentase\ Error = \frac{Percobaan\ Gagal}{Banyak\ percobaan} \times 100\ \% \quad (6.1)$$

$$Presentase\ Error = \frac{0}{10} \times 100\% = 0\ \%$$

6.2 Pengujian Ketepatan Waktu Estimasi Dengan Waktu Sebenarnya

6.2.1 Tujuan Pengujian

Tujuan dari pengujian ketepatan waktu estimasi dengan waktu sebenarnya adalah untuk melihat seberapa besar kemiripan dan seberapa besar *error* yang dihasilkan. Keberhasil dan *error* didasarkan pada selisih nilai antara waktu estimasi dan waktu terbang.

6.2.2 Pelaksanaan Pengujian

Pengujian dilaksanakan dengan membandingkan waktu terbang dan waktu estimasi. Waktu estimasi didapat dari perhitungan yang didasarkan pada kapasitas baterai, sedangkan waktu terbang dilakukan dengan menggunakan *stopwatch*.

6.2.3 Prosedur Pengujian

1. Menyalakan *quadcopter*, dan pastikan *quadcopter* tidak dalam mode *emergency*(led berwarna merah).

2. Memastikan kapasitas baterai lebih dari 75 %.

3. Menyambungkan *quadcopter* ke komputer menggunakan wifi.

4. Membuka *Command Line Interface* untuk Node JS.

5. Memasuki direktori program utama.

6. Mengeksekusi program utama dengan syntax “node “nama_file”.js”

7. Tunggu hingga program menampilkan *feedback* pada *node js console terminal*

8. Membuka browser dan panggil menggunakan *port* yang digunakan.

9. Menerbangkan *quadcopter* dan mulai stopwatch.

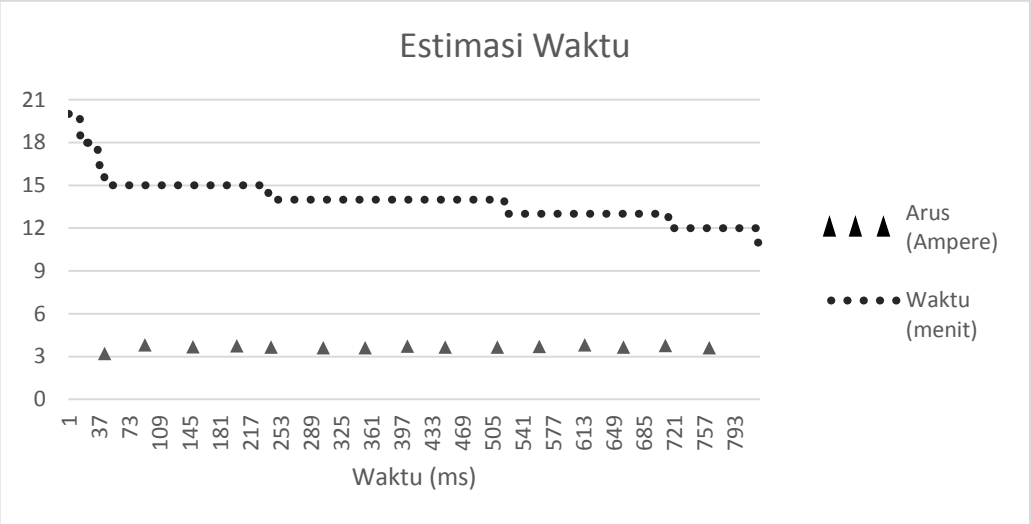
10. Mengamati waktu estimasi

11. Ketika presentase baterai menujukkan 15% amati waktu estimasi yang tersisa.

12. Jika benar ketika menunjukkan 15 % waktu estimasi bernilai kurang dari 1 menit.

6.2.4 Hasil Pengujian Ketepatan Waktu Estimasi Dengan Waktu Sebenarnya

Keberhasilan dari pengujian ini bergantung pada selisih nilai antara waktu estimasi dan waktu terbang *quadcopter,* semakin sedikit nilai selisihnya maka semakin kecil tingkat errornya, sehingga Ketepatan nilai akan semakin tinggi. Pengujian dilakukan sebanyak 10 kali pada setiap gerakan dan dengan *input* kecepatan sebesar 03 hingga 0.7.



Gambar 6.3 Grafik Hubungan Estimasi Waktu dan Arus total

Gambar 6.3 menunjukkan grafik hubungan antara nilai output dari estimasi waktu dan arus total dari *quadcopter*. Total arus ditunjukkan dengan bentuk segitiga yang dinyatakan dalam satuan ampere. Dan estimasi waktu ditunjukkan dengan bentuk titik yang dinyatakan dalam satuan menit.

6.2.5 Analisis Pengujian Ketepatan Waktu Estimasi Dengan Waktu Sebenarnya

Dari pengujian yang dilakukan sebanyak 7 kali didapat data berupa waktu terbang *quadcopter* dan selisih waktu yang diketahui dengan menggunakan *stopwatch*. Nilai kesalahan didapat dari selisih estimasi waktu dengan waktu terbang *quadcopter*, dan untuk mengetahui presentase kesalahan dilakukan perhitungan sesuai dengan persamaan 6.3.

Tabel 6.2 Analisa Pengujian

Gerakan	*Error* (%)					
	Input kecepatan					Rata-rata
	0.3	0.4	0.5	0.6	0.7	
Hover	1	1	0.7	0.8	1.2	0.94
Maju	1.1	1.2	1.1	1.5	1.1	1.2
Mundur	1.5	1.7	1	1.2	1.5	1.38
Kiri	1.1	1.3	0.7	1.1	1.5	1.14
Kanan	1.2	1	0.9	1.3	1	1.08
Total Error						1.148

Tabel 6.2 merupakan tabel hasil dari pengujian ketepatan estimasi waktu yang berisi nilai kesalahan estimasi waktu. Kesalahan / *error* yang di tampilkan merupkana error dari rata-rata dari gerakan dan input kecepatan sebanyak 10 kali. Hasil rata-rata tersebut kemudian di rata-rata kembali. Sehingga didapat total error dari setiap gerakan. Untuk menghitungan total *error* digunakan perhitungan menggunakan persamaan 6.2 dan 6.3

$$\overline{Gerakan} = \frac{\sum_0^5 u_i}{5} \tag{6.2}$$

$$Hover = {}^{1 + 1 + 0.7 + 0.8 + 1.2}/_{5} = 0.94\ \%$$

$$Maju = {}^{1.1 + 1.2 + 1.1 + 1.5 + 1.1}/_{5} = 1.2\ \%$$

$$Mundur = {}^{1.5 + 1.7 + 1 + 1.2 + 1.5}/_{5} = 1.38\ \%$$

$$Kiri = {}^{1.1 + 1.3 + 0.7 + 1.1 + 1.5}/_{5} = 1.14\ \%$$

$$Kanan = {}^{1.2 + 1 + 0.9 + 1.3 + 1}/_{5} = 1.08\ \%$$

$$\overline{total\ error} = {}^{\sum_{0.9}^{5} a_i}/_{5} \tag{6.3}$$

$$\overline{total\ error} = {}^{0.94 + 1.2 + 1.38 + 1.14 + 1.08}/_{5} = 1.148\ \%$$

Persamaan 6.2 adalah persamaan yang digunakan untuk menghitung arata rata kesalahan pada tiap gerakan, sedangkan persamaan 6.3 merupakan persamaan yang digunakan untuk menghitung total error pada ketepatan estimasi waktu. Dengan batas bawah adalah 1, batas atas adalah 5, dan $\boldsymbol{a_i}$ adalah suku yang merepresentasikan rata-rata *error* dari pengujian pada gerakan yang disebutkan.

6.3 Pengujian Ketepatan Perpindahan *Quadcopter*

6.3.1 Tujuan Pengujian

Tujuan dari pengujian ketepatan perpindahan *quadcopter* adalah untuk mengetahui seberapa akurat data yang ditampilkan pada antarmuka *web* dengan perpindahan *quadcopter* dari titik awalnya. Dalam analisis pengujian ini dilakukan pemantauan seberapa besar kesalah dari estimasi perpindahan *quadcopter* dari titik awalnya.

6.3.2 Pelaksanaan Pengujian

Pengujian dilaksanakan dengan cara menerbangkan *quadcopter* selama satu menit, dan melihat seberapa jauh perpindahan yang dapat di lakukan oleh *quadcopter*. Perpindahan diukur menggunakan meteran, dan dengan menggunakan *stopwatch* untuk dapat mengetahui bahwa *quadcopter* telah terbang selama 1 menit. Pengujian dilakukan pada operasi maju, mundur, ke kiri dan ke kanan dengan *input* kecepatan yang berbeda - beda.

6.3.3 Prosedur Pengujian

1. Menyalakan *quadcopter*, dan memastikan *quadcopter* tidak dalam mode *emergency*(*led* berwarna merah).
2. Memastikan kapasitas baterai lebih dari 75 %
3. Menyambungkan *quadcopter* ke komputer menggunakan *wifi*.
4. Membuka *Console terminal Node JS*.
5. Memasuki kedalam direktori program utama.
6. Mengeksekusi program utama dengan syntax “node “nama_file”.js”.
7. Menunggu *feedback* yang ditampilkan pada *node js console terminal*.
8. Membuka *browser* dan memanggil menggunakan *port* yang digunakan.
9. Menerbangkan *quadcopter* dan memulai *stopwatch*.
10. Setelah 1 menit, melakukan *landing* pada *quadcopter* , dan mengukur perpindahan dari titik awal hingga tempat saat *quadcopter* mendarat.

6.3.4 Hasil Pengujian

Tabel 6.3 menunjukkan hasil dari pengujian terhadap ketepatan perpindahan *quadcopter*. Tabel 6.3 memuat informasi berupa perpindahan *quadcopter* selama satu menit dengan *input* kecepatan mulai dari 0.3 hingga 0.7 dengan gerakan maju, mundur, gerak ke kiri, dan gerak ke kanan.

Tabel 6.3 Pengujian Perpindahan *Quadcopter*

Input kecepatan	Perpindahan			
	Maju	Mundur	Ke Kiri	Ke Kanan
0.3	22 m	30 m	26 m	25 m
0.4	24 m	25 m	23 m	21 m
0.5	35 m	32 m	26 m	28 m
0.6	51 m	43 m	56 m	40 m
0.7	49 m	51 m	51 m	49 m

6.3.5 Analisis Pengujian perpindahan *Quadcopter*

Tabel 6.4 memuat informasi mengenai presentase kesalahan estimasi waktu. Tabel 6.4 menunjukkan presentase kesalahan pada gerak maju, mundur, ke kiri dan ke kanan dengan input kecepatan sebesar 0.3 hingga 0.7. Nilai error dari setiap gerakan kemudian di rata-rata, dan kemudian di rata-rata kembali hingga didapat nilai *error* total. Penghitungan *error* dilakukan dengan menggunakan persamaan 6.4, 6.5, dan 6.6.

Tabel 6.4 *Error* Estimasi *Quadcopter*

	Rata-rata kesalahan (%)					
	Input kecepatan					Rata-rata
Gerakan	0.3	0.4	0.5	0.6	0.7	
Maju	37.7	32.1	0.9	44.2	38.6	30.7
Mundur	15.1	29.2	9.4	21.6	44.2	23.9
Ke Kiri	26.4	34.9	26.4	58.4	44.2	38.0
Ke Kanan	29.2	29.2	20.7	13.1	38.6	26.2
Total Error						29.7

$$Error\ Estimasi\ perpindahan = \frac{|perpindahan - selisih|}{selisih} \times 100\ \% \qquad (6.4)$$

	Maju		Mundur
0.3	$\frac{\|35.35 - 22\|}{22} \times 100\% = 37.7$	0.3	$\frac{\|35.35 - 30\|}{30} \times 100\% = 15.1$
0.4	$\frac{\|35.35 - 24\|}{24} \times 100\% = 32.1$	0.4	$\frac{\|35.35 - 25\|}{25} \times 100\% = 29.2$
0.5	$\frac{\|35.35 - 35\|}{35} \times 100\% = 0.9$	0.5	$\frac{\|35.35 - 32\|}{32} \times 100\% = 9.4$
0.6	$\frac{\|35.35 - 51\|}{51} \times 100\% = 44.2$	0.6	$\frac{\|35.35 - 43\|}{43} \times 100\% = 21.6$
0.7	$\frac{\|35.35 - 49\|}{49} \times 100\% = 38.6$	0.7	$\frac{\|35.35 - 51\|}{51} \times 100\ \% = 44.2$
	Ke Kiri		Ke Kanan
0.3	$\frac{\|35.35 - 26\|}{22} \times 100\ \% = 26.4$	0.3	$\frac{\|35.35 - 25\|}{30} \times 100\% = 29.2$

0.4	$\frac{\lvert 35.35-23\rvert}{24}\times 100\ \% = 34.9$	0.4	$\frac{\lvert 35.35-21\rvert}{25}\times 100\% = 29.2$
0.5	$\frac{\lvert 35.35-36\rvert}{35}\times 100\ \% = 26.4$	0.5	$\frac{\lvert 35.35-28\rvert}{32}\times 100\% = 20.7$
0.6	$\frac{\lvert 35.35-56\rvert}{51}\times 100\ \% = 58.4$	0.6	$\frac{\lvert 35.35-40\rvert}{43}\times 100\% = 13.15$
0.7	$\frac{\lvert 35.35-51\rvert}{49}\times 100\ \% = 44.2$	0.7	$\frac{\lvert 35.35-49\rvert}{51}\times 100\% = 38.61$

$$\overline{n\ Input\ Kecepatan} = \frac{\sum_0^4 u_i}{4} \quad (6.5)$$

$$input\ 0.3 = {}^{37.76+15.13+26.44+29.27}/_{4} = 27.15\ \%$$

$$input\ 0.4 = {}^{32.12+29.93+34.93+29.27}/_{4} = 31.42\ \%$$

$$input\ 0.5 = {}^{00.99+09.47+26.44+20.79}/_{4} = 14.42\ \%$$

$$input\ 0.6 = {}^{44.27+21.64+58.41+13.15}/_{4} = 34.37\ \%$$

$$input\ 0.7 = {}^{38.61+44.27+44.27+38.61}/_{4} = 41.44\ \%$$

$$\overline{total\ error} = {}^{\sum_{0.3}^{5} a_i}/_{5} \quad (6.6)$$

$$\overline{total\ error} = {}^{27.15+31.42+14.42+34.37+41.44}/_{5} = 29.75\ \%$$

Persamaan 6.2 adalah persamaan yang digunakan untuk mengihtung presentase kesalahan pada input kecepatan mulai 0.3 hingga 0.7 Dan persamaan 6.3 merupakan persamaan yang digunakan untuk menghitung total error pada ketepatan estimasi waktu. Dengan batas bawah adalah 1, batas atas adalah 5, dan $\boldsymbol{a_i}$ adalah suku yang merepresentasikan rata-rata *error* dari pengujian pada gerakan yang disebutkan.

Dari analisis yang dilakukan pada pengujian ketepatan perpindahan yang ditunjukkan tabel 6.4 yang berisi nilai *error* pada setiap gerakan dengan *input* kecepatan input berbeda-beda. Selanjutnya dari data yang didapat kemudian dihitung dengan menggunakan persamaan (6.4), (6.5), dan persamaan (6.5). Persamaan 6.13 digunakan untuk menghitung error pada setiap data yang didapatkan, pada persamaan 6.34 digunakan untuk menghitung rata-rata *error* pada setiap input kecepatan mulia dari 0.3 – 0.7, sedangkan persamaan 6.4 digunakan untuk menghitung nilai rata-rata *error* total. Sehingga dari pengujian yang dilakukan didapat hasil *error* sebesar 29.75 %

BAB 7 PENUTUP

7.1 Kesimpulan

Dari hasil analisis data yang telah dilakukan, maka didapat kesimpulan dari apa yang telah dilakukan dari bab sebelumnya. Kesimpulan yang didapat antara lain adalah :

1. Perhitungan kapasitas dan arus komponen pada AR.Drone 2.0 yang dibuat dengan tujuan agar kinerja *quadcopter* menjadi lebih optimal yang dibuat dengan bahasa pemrograman *javascript* dan dengan menggunakan antarmuka berupa web berhasil diimplementasikan dan menunjukkan hasil yang cukup baik dengan *error* yang relatif kecil.
2. Pengendalian dengan menggunakan keyboard laptop menujukkan hasil yang sangat baik karena kesalahan pengendalian adalah 0%
3. Estimasi waktu pada saat *quadcopter* melakukan operasi terbang menunjukkan hasil yang baik dengan tingkat kesalahan yang didapat sangat kecil yaitu 1.148 %
4. Estimasi perpindahan yang dilakukan pada saat *quadcopter* terbang menunjukkan hasil yang cukup baik dengan tingkat kesalahan yang dihasilkan adalah sebesar +/- 29 %

7.2 Saran

Dari hasil yang telah selesai di uji dan di analisis, sehingga didapatkan hasil yang cukup baik dari sistem. Namun terdapat kekurangan dalam sistem, yang mungkin pada proses pengembangan berikutnya dapat lebih diperbaiki lagi, adapun hal yang menjadi saran untuk penelitian berikutnya antara lain adalah.

1. Pengujian tentang ketepatan gerakan yang menujukkan nilai keberhasilan hingga 100%. Sistem dapat dikombinasikan pada alternatif pengendalian seperti pada kinect, leap motion, google cardboard, dan pada aplikasi pengendalian pada *smartphone*.
2. *Error* yang didapat dari estimasi waktu masih cukup besar, sehingga perlu diterapkan sebuah algoritma untuk dapat mengkoreksi apabila terjadi ketidak sesuaian estimasi saat mendekati sisa baterai yang direkomendasikan.
3. Nilai estimasi perpindahan didapat dari hasil percobaan terbang sehingga didapat besar perpindahan per satuan waktu. Akan lebih baik apabila perpindahan didapat dengan menggunakan rumus perpindahan yang diperoleh dari rumus kecepatan linear *quadcopter*. Namun diperlukan data kecepatan dengan nilai yang stabil. Dan untuk menstabilkan nilai kecepatan dari *quadcopter* dapat dilakukan dengan metode statistik, dengan memanfaatkan regresi linear real time, sehingga estimasi perpindahan bisa didapatkan secara realtime dan aktual.

DAFTAR PUSTAKA

ALbert, M. & David, J., 2006. *Electronic Principles,(7th Edition).* Standford: McGraw-Hill Education.

Albert, A., 2014. *Javascript Lanjut.* [Online] Available at: https://bertzzie.com/[Accessed 14 June 2017].

Buchmann, I., 2017. *Battery University.* [Online]Availablat:http://batteryuniversity.com /learn/article/how_to_measure_internal_resistance [Accessed 4 April 2017].

Darsiwan, 2016. *CodePolitan.* [Online] Available at: https://www.codepolitan.com/ menegtahui-apa-itu-websocket[Accessed 6 August 2017].Devnull, 2012.*https:// www.libcrack.so.* [Online] Available at: https://www.libcrack.so/2012/10/13/ hacking-the-ar-drone-parrot/ [Accessed 28 March 2017].

ExAir, 2012. *https://www.flitetest.com.* [Online] Available at:https://www.flitetest.com/ articles/LiPo_Battery_Internal_Resistance_Testing [Accessed 28 March 2017].

Fastly, 2017. *Epoch.* [Online] Available at: https://epochjs.github.io/epoch/[Accessed 6 May 2017].

Ghazbi, N., 2016. Quadrotors Unmanned Aerial Vehicles: A Review. *International Journal On Smart Sensing And Intelegent Systems,* Volume IX, p. 25.

Hadi, S W., Setyawan, G E. & Maulana, R., 2018. Sistem Kendali Navigasi Ar.Drone Quadcopter Dengan Prinsip Natural. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK),* Volume II, pp. 380-386.

Jose, M., 2011. *Vision Path Following with a Stabilized Quadrotor.* 1st ed. Lisboa: Instituto Superior Tecnico.

Kuphaldt, T., 2006. *Lessons In Electric Circuits, Volume I – DC.* Bellingham: Design Scinece License.

Malvino, A., 2006. *Electronic Principles.* 7th ed. Stanford: McGraw-Hill Education.

Maulana, E., 2012. *Pengaturan PWM dengan PLC.* 1st ed. Malang: Fakultas Teknik.

Muttaqin, A., Prasetio, B H. & Setiawan, E., 2012. *Sinyal Rangkaian.* 1st ed. Malang: Program Teknologi Informasi dan Ilmu Komputer.

Niko, 2014. *Pintar Komputer.* [Online] Available at: http://www.pintarkomputer.com/ [Accessed 5 August 2017].

Pallas, F A., Setyawan, G E. & Prasetio, B H., 2018. Sistem Kendali Navigasi Quadcopter Menggunakan Suara Melalui. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK),* Volume II, pp. 732-738.

Pena, K., 2014. Accountability for Private Security Contractor Drone Operators on the U.S.-Mexico Border: Applying Lessons Learned from the Middle East. *Public Contract Law Journal,* Volume XLIV, p. 137.

Piskroski, S., Brulez, N., Eline, P. & D'Haeyer, F., 2012. *AR.Drone Developer Guide Revision SDK 2.0.* San Andreas: Parrot.

Pribadi, D A., Jonimaro, E M A. & Setyawan, G E., 2017. Implementasi Pengendalian Quadcopter Dengan Prinsip Virtual Reality Menggunakan Google Cardboard. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK),* Volume I, pp. 1451-1458.

Rabaca, M. J. J., 2011. *Vision Path Following with a Stabilized Quadrotor.* Lisboa: Instituto Superior Tecnico.

Sarkar, A., Patel, K., Ram, G. & Capoor, K., 2016. Gesture Control of Drone Using A Motion Controller. *ICCSII,* pp. 1-5.

Setyawan, G E., Setyawan, E. & Kurniawan, W., 2015. Sistem Kendali Ketinggian Quadcopter Menggunakan PID. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK),* Volume II, pp. 125-131.

Turo, T., 2015. Monitoring of Military Vehicel Battery. *IEEE,* pp. 1-4.

Vechian, M., 2012. *Wireless Control Quadcopter With Stereo Camera And Self-Balancing System.* Johor: s.n.

W3.CSS, 2017. *W3Schools.com.* [Online] Available at:https://www.w3schools.com/nodejs/nodejs_filesystem.asp[Accessed 14 August 2017].

Williard, N., 2011. Predicting Remaining Capacity of Batteries for UAVs and Electric Vehicle. *IMAPS Advanced Technology Workshop on High Reliability Microelectronics for Military Applications,* pp. 17-19.

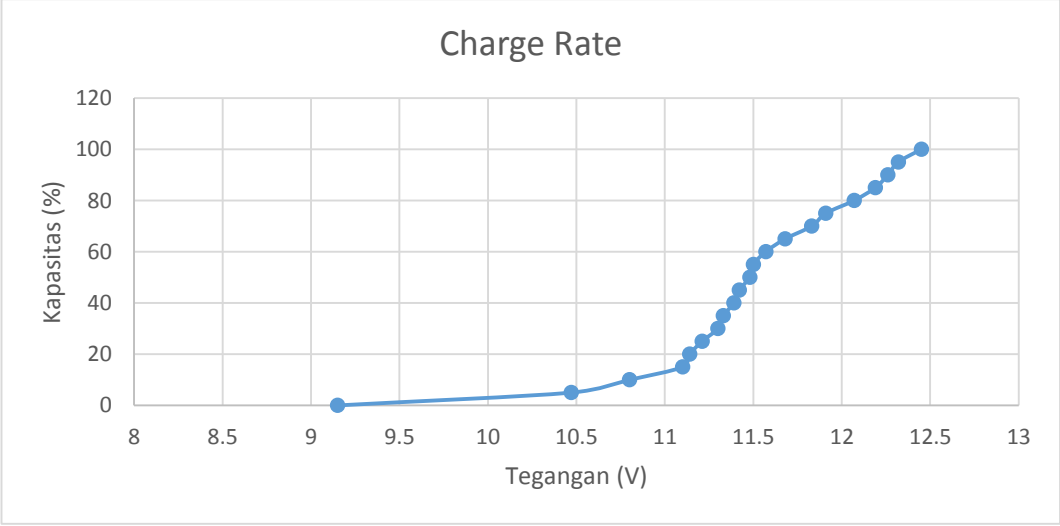
LAMPIRAN

a. Charging state

Tabel *Charging state*

Kapasitas (%)	Tegangan (V)
100	12.45
95	12.32
90	12.26
85	12.19
80	12.07
75	11.91
70	11.83
65	11.68
60	11.57
55	11.5
50	11.48
45	11.42
40	11.39
35	11.33
30	11.3
25	11.21
20	11.14
15	11.1
10	10.8
5	10.47
0	9.15

Gambar Grafik Charging State



b. Pengujian Ketepatan Estimasi Waktu

Pengujian Ketepatan Estimasi Waktu Dengan input kecepatan 0.3

Nomor	Gerak	Kapasitas Baterai	Waktu	*web*	*start*	*end*	Waktu (s)	*Error*
1	*Hover*	1500 mAh	11:15	11:00	96%	15%	675	2.272727
2	*Hover*	1500 mAh	11:11	11:00	96%	15%	671	1.666667
3	*Hover*	1500 mAh	11:02	11:00	96%	15%	662	0.30303
4	*Hover*	1500 mAh	11:05	11:00	96%	15%	665	0.757576
5	*Hover*	1500 mAh	10:51	11:00	96%	15%	651	1.363636
6	*Hover*	1500 mAh	10:49	11:00	96%	15%	649	1.666667
7	*Hover*	1500 mAh	10:58	11:00	96%	15%	658	0.30303
8	*Hover*	1500 mAh	11:12	11:00	96%	15%	672	1.818182
9	*Hover*	1500 mAh	10:59	11:00	96%	15%	659	0.151515
10	*Hover*	1500 mAh	11:00	11:00	96%	15%	660	0
Total Error								1.030303

Nomor	Gerak	Kapasitas Baterai	Waktu	*Web*	*Start*	*End*	Waktu (s)	*Error*
1	*Forward*	1500 mAh	6:13	6:00	53%	15%	363	0.833333
2	*Forward*	1500 mAh	6:19	6:00	53%	15%	364	1.111111
3	*Forward*	1500 mAh	6.01	6:00	53%	15%	361	0.277778
4	*Forward*	1500 mAh	6:10	6:00	53%	15%	367	1.944444
5	*Forward*	1500 mAh	5:58	6:00	53%	15%	358	0.555556
6	*Forward*	1500 mAh	5:57	6:00	53%	15%	357	0.833333
7	*Forward*	1500 mAh	5:54	6:00	53%	15%	354	1.666667
8	*Forward*	1500 mAh	6:07	6:00	53%	15%	367	1.944444
9	*Forward*	1500 mAh	6:09	6:00	53%	15%	359	0.277778
10	*Forward*	1500 mAh	5:51	6:00	53%	15%	353	1.944444
Total Error								1.138889

Nomor	Gerak	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	*Error*
1	Backward	1500 mAh	5:55	6:00	50%	15%	355 s	1.388889
2	Backward	1500 mAh	5:51	6:00	50%	15%	351 s	2.5
3	Backward	1500 mAh	5:54	6:00	50%	15%	354 s	1.666667
4	Backward	1500 mAh	5:52	6:00	50%	15%	352 s	2.222222
5	Backward	1500 mAh	5:59	6:00	50%	15%	359 s	0.277778
6	Backward	1500 mAh	5:52	6:00	50%	15%	352 s	2.222222
7	Backward	1500 mAh	6:00	6:00	50%	15%	360 s	0
8	Backward	1500 mAh	6:05	6:00	50%	15%	365 s	1.388889
9	Backward	1500 mAh	5:53	6:00	50%	15%	353 s	1.944444
10	Backward	1500 mAh	5:52	6:00	50%	15%	352 s	2.222222

Total Error								1.583333
Nomor	Gerak	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	*Error*
1	Go Left	1500 mAh	6:52	7:00	65%	15%	412	1.904762
2	Go Left	1500 mAh	6:51	7:00	65%	15%	411	2.142857
3	Go Left	1500 mAh	6:59	7:00	65%	15%	419	0.238095
4	Go Left	1500 mAh	7:01	7:00	65%	15%	421	0.238095
5	Go Left	1500 mAh	7:05	7:00	65%	15%	425	1.190476
6	Go Left	1500 mAh	7:03	7:00	65%	15%	423	0.714286
7	Go Left	1500 mAh	6:55	7:00	65%	15%	415	1.190476
8	Go Left	1500 mAh	6:52	7:00	65%	15%	412	1.904762
9	Go Left	1500 mAh	7:12	7:00	65%	15%	420	0
10	Go Left	1500 mAh	7:09	7:00	65%	15%	429	2.142857
Total Error								1.166667

Nomor	Gerak	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	*Error*
1	Go right	1500 mAh	8:55	9:00	80%	15%	535	0.925926
2	Go right	1500 mAh	9:12	9:00	80%	15%	552	2.222222
3	Go right	1500 mAh	9:10	9:00	80%	15%	550	1.851852
4	Go right	1500 mAh	9:05	9:00	80%	15%	545	0.925926
5	Go right	1500 mAh	9:01	9:00	80%	15%	541	0.185185
6	Go right	1500 mAh	9:05	9:00	80%	15%	545	0.925926
7	Go right	1500 mAh	9:11	9:00	80%	15%	551	2.037037
8	Go right	1500 mAh	8:57	9:00	80%	15%	537	0.555556
9	Go right	1500 mAh	9:09	9:00	80%	15%	549	1.666667
10	Go right	1500 mAh	8:52	9:00	80%	15%	532	1.481481
Total Error								1.277778

Pengujian Ketepatan Estimasi Waktu Dengan input kecepatan 0.4

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu(S)	Error
1	*Hover*	1500 mAh	11:12	11:00	90%	15%	672	1.818182
2	*Hover*	1500 mAh	11:08	11:00	90%	15%	668	1.212121
3	*Hover*	1500 mAh	10:59	11:00	90%	15%	659	0.151515
4	*Hover*	1500 mAh	10:02	11:00	90%	15%	662	0.30303
5	*Hover*	1500 mAh	10:48	11:00	90%	15%	648	1.818182
6	*Hover*	1500 mAh	10:46	11:00	90%	15%	646	2.121212
7	*Hover*	1500 mAh	10:55	11:00	90%	15%	655	0.757576
8	*Hover*	1500 mAh	11:09	11:00	90%	15%	669	1.363636
9	*Hover*	1500 mAh	11:06	11:00	90%	15%	666	0.909091
10	*Hover*	1500 mAh	10:57	11:00	90%	15%	657	0.454545
Total Error								1.090909

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu (S)	Error
1	*Forward*	1500 mAh	6:02	6:00	48%	15%	362	0.555556
2	*Forward*	1500 mAh	6:02	6:00	48%	15%	362	0.555556
3	*Forward*	1500 mAh	5:59	6:00	48%	15%	359	0.277778
4	*Forward*	1500 mAh	6:05	6:00	48%	15%	365	1.388889
5	*Forward*	1500 mAh	5:56	6:00	48%	15%	356	1.111111
6	*Forward*	1500 mAh	5:55	6:00	48%	15%	355	1.388889
7	*Forward*	1500 mAh	5:52	6:00	48%	15%	352	2.222222
8	*Forward*	1500 mAh	6:02	6:00	48%	15%	365	1.388889
9	*Forward*	1500 mAh	5:57	6:00	48%	15%	357	0.833333
10	*Forward*	1500 mAh	5:51	6:00	48%	15%	351	2.5
Total Error								1.222222

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu (S)	Error
1	Backward	1500 mAh	5:53	6:00	48%	15%	353	1.9444
2	Backward	1500 mAh	5:54	6:00	48%	15%	354	1.6666
3	Backward	1500 mAh	5:53	6:00	48%	15%	353	1.9444
4	Backward	1500 mAh	5:51	6:00	48%	15%	351	2.5
5	Backward	1500 mAh	5:58	6:00	48%	15%	358	0.5555
6	Backward	1500 mAh	5:01	6:00	48%	15%	351	2.5
7	Backward	1500 mAh	5:58	6:00	48%	15%	358	0.5555
8	Backward	1500 mAh	6:03	6:00	48%	15%	363	0.8333
9	Backward	1500 mAh	5:52	6:00	48%	15%	352	2.2222
10	Backward	1500 mAh	5:50	6:00	48%	15%	350	2.7777
Total Error								1.75

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu(S)	Error
1	*Go left*	1500 mAh	6:50	7:00	60%	15%	410	2.3809
2	*Go left*	1500 mAh	6:49	7:00	60%	15%	409	2.6190
3	*Go left*	1500 mAh	6:57	7:00	60%	15%	417	0.7142
4	*Go left*	1500 mAh	6:59	7:00	60%	15%	419	0.2380
5	*Go left*	1500 mAh	7:03	7:00	60%	15%	423	0.7142
6	*Go left*	1500 mAh	7:01	7:00	60%	15%	421	0.2380
7	*Go left*	1500 mAh	6:53	7:00	60%	15%	413	1.6666
8	*Go left*	1500 mAh	6:50	7:00	60%	15%	410	2.3809
9	*Go left*	1500 mAh	6:58	7:00	60%	15%	418	0.4761
10	*Go left*	1500 mAh	7:07	7:00	60%	15%	427	1.6666
Total Error								1.3095

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	Error
1	*Go right*	1500 mAh	8:53	9:00	78%	15%	533 s	1.2962
2	*Go right*	1500 mAh	9:10	9:00	78%	15%	550 s	1.8518
3	*Go right*	1500 mAh	9:08	9:00	78%	15%	548 s	1.4814
4	*Go right*	1500 mAh	9:03	9:00	78%	15%	543 s	0.5555
5	*Go right*	1500 mAh	8:59	9:00	78%	15%	539 s	0.1851
6	*Go right*	1500 mAh	9:03	9:00	78%	15%	543 s	0.5555
7	*Go right*	1500 mAh	9:03	9:00	78%	15%	543 s	0.5555
8	*Go right*	1500 mAh	8:55	9:00	78%	15%	535 s	0.9259
9	*Go right*	1500 mAh	9:07	9:00	78%	15%	547 s	1.2962
10	*Go right*	1500 mAh	8:50	9:00	78%	15%	530 s	1.8518
Total Error								1.0555

Pengujian Ketepatan Estimasi Waktu Dengan input kecepatan 0.5

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	*Error*
1	*Hover*	1500 mAh	10:01	10:00	85%	15%	601 s	0.1666
2	*Hover*	1500 mAh	10:04	10:00	85%	15%	604 s	0.6666
3	*Hover*	1500 mAh	10:09	10:00	85%	15%	609 s	1.5
4	*Hover*	1500 mAh	9:57	10:00	85%	15%	597 s	0.5
5	*Hover*	1500 mAh	9:59	10:00	85%	15%	599 s	0.1666
6	*Hover*	1500 mAh	10:07	10:00	85%	15%	607 s	1.1666
7	*Hover*	1500 mAh	10:03	10:00	85%	15%	603 s	0.5
8	*Hover*	1500 mAh	9:53	10:00	85%	15%	593 s	1.1666
9	*Hover*	1500 mAh	10:10	10:00	85%	15%	610 s	1.6666
10	*Hover*	1500 mAh	9:58	10:00	85%	15%	598 s	0.3333
Total Error								0.7833

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	*Error*
1	*Forward*	1500 mAh	6:59	7:00	62%	15%	419 s	0.2380
2	*Forward*	1500 mAh	7:03	7:00	62%	15%	423 s	0.7142
3	*Forward*	1500 mAh	7:09	7:00	62%	15%	429 s	2.1428
4	*Forward*	1500 mAh	7:11	7:00	62%	15%	431 s	2.6190
5	*Forward*	1500 mAh	7:07	7:00	62%	15%	427 s	1.6666
6	*Forward*	1500 mAh	6:55	7:00	62%	15%	415 s	1.1904
7	*Forward*	1500 mAh	7:05	7:00	62%	15%	425 s	1.1904
8	*Forward*	1500 mAh	7:04	7:00	62%	15%	424 s	0.9523
9	*Forward*	1500 mAh	6:56	7:00	62%	15%	416 s	0.9523
10	*Forward*	1500 mAh	7:01	7:00	62%	15%	421 s	0.2380
Total Error								1.1904

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	*Error*
1	*Backward*	1500 mAh	7:07	7:00	62%	15%	427 s	1.666
2	*Backward*	1500 mAh	7:01	7:00	62%	15%	421 s	0.238
3	*Backward*	1500 mAh	7:03	7:00	62%	15%	423 s	0.714
4	*Backward*	1500 mAh	7:04	7:00	62%	15%	424 s	0.952
5	*Backward*	1500 mAh	6:56	7:00	62%	15%	416 s	0.952
6	*Backward*	1500 mAh	6:59	7:00	62%	15%	419 s	0.238
7	*Backward*	1500 mAh	7:06	7:00	62%	15%	427 s	1.666
8	*Backward*	1500 mAh	7:02	7:00	62%	15%	422 s	0.476
9	*Backward*	1500 mAh	6:52	7:00	62%	15%	412 s	1.904
10	*Backward*	1500 mAh	6:54	7:00	62%	15%	414 s	1.428
Total Error								1.023

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	*Error*
1	*Go Left*	1500 mAh	7:03	7:00	62%	15%	423 s	0.714
2	*Go Left*	1500 mAh	7:04	7:00	62%	15%	424 s	0.952
3	*Go Left*	1500 mAh	7:02	7:00	62%	15%	422 s	0.476
4	*Go Left*	1500 mAh	7:00	7:00	62%	15%	420 s	0
5	*Go Left*	1500 mAh	6:55	7:00	62%	15%	415 s	1.190
6	*Go Left*	1500 mAh	6:58	7:00	62%	15%	418 s	0.476
7	*Go Left*	1500 mAh	6:57	7:00	62%	15%	417 s	0.714
8	*Go Left*	1500 mAh	6:59	7:00	62%	15%	419 s	0.238
9	*Go Left*	1500 mAh	7:04	7:00	62%	15%	424 s	0.952
10	*Go Left*	1500 mAh	7:09	7:00	62%	15%	429 s	2.142
Total Error								0.785

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	*Error*
1	*Go Right*	1500 mAh	9:11	9:00	73%	15%	551 s	2.037
2	*Go Right*	1500 mAh	9:09	9:00	73%	15%	549 s	1.666
3	*Go Right*	1500 mAh	9:03	9:00	73%	15%	543 s	0.555
4	*Go Right*	1500 mAh	9:04	9:00	73%	15%	544 s	0.740
5	*Go Right*	1500 mAh	9:07	9:00	73%	15%	547 s	1.296
6	*Go Right*	1500 mAh	8:53	9:00	73%	15%	533 s	1.296
7	*Go Right*	1500 mAh	8:59	9:00	73%	15%	539 s	0.185
8	*Go Right*	1500 mAh	9:02	9:00	73%	15%	542 s	0.370
9	*Go Right*	1500 mAh	8:55	9:00	73%	15%	535 s	0.925
10	*Go Right*	1500 mAh	8:57	9:00	73%	15%	537 s	0.555
Total Error								0.962

Pengujian Ketepatan Estimasi Waktu Dengan input kecepatan 0.6

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	*Error*
1	*Hover*	1500 mAh	9:11	9:00	80%	15%	551 s	2.0370
2	*Hover*	1500 mAh	9:02	9:00	80%	15%	542 s	0.3703
3	*Hover*	1500 mAh	9:01	9:00	80%	15%	541 s	0.1851
4	*Hover*	1500 mAh	9:05	9:00	80%	15%	545 s	0.9259
5	*Hover*	1500 mAh	9:02	9:00	80%	15%	542 s	0.3703
6	*Hover*	1500 mAh	9:07	9:00	80%	15%	547 s	1.2962
7	*Hover*	1500 mAh	8:55	9:00	80%	15%	535 s	0.9259
8	*Hover*	1500 mAh	8:57	9:00	80%	15%	537 s	0.5555
9	*Hover*	1500 mAh	9:03	9:00	80%	15%	543 s	0.5555
10	*Hover*	1500 mAh	9:05	9:00	80%	15%	545 s	0.9259
Total Error								0.8148

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	*Error*
1	*Forward*	1500 mAh	6:03	6:00	58%	15%	363 s	0.833
2	*Forward*	1500 mAh	6:08	6:00	58%	15%	368 s	2.222
3	*Forward*	1500 mAh	6:02	6:00	58%	15%	362 s	0.555
4	*Forward*	1500 mAh	6:11	6:00	58%	15%	371 s	3.055
5	*Forward*	1500 mAh	6:05	6:00	58%	15%	365 s	1.388
6	*Forward*	1500 mAh	6:07	6:00	58%	15%	367 s	1.944
7	*Forward*	1500 mAh	5:55	6:00	58%	15%	355 s	1.388
8	*Forward*	1500 mAh	5:57	6:00	58%	15%	357 s	0.833
9	*Forward*	1500 mAh	6:01	6:00	58%	15%	361 s	0.277
10	*Forward*	1500 mAh	6:10	6:00	58%	15%	370 s	2.777
Total Error								1.527

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	*Error*
1	*Backward*	1500 mAh	5:57	6:00	58%	15%	357 s	0.833
2	*Backward*	1500 mAh	6:05	6:00	58%	15%	365 s	1.388
3	*Backward*	1500 mAh	6:06	6:00	58%	15%	363 s	0.833
4	*Backward*	1500 mAh	6:00	6:00	58%	15%	360 s	0
5	*Backward*	1500 mAh	6:05	6:00	58%	15%	365 s	1.388
6	*Backward*	1500 mAh	5:56	6:00	58%	15%	356 s	1.111
7	*Backward*	1500 mAh	5:53	6:00	58%	15%	353 s	1.944
8	*Backward*	1500 mAh	6:12	6:00	58%	15%	372 s	3.333
9	*Backward*	1500 mAh	5:59	6:00	58%	15%	359 s	0.277
10	*Backward*	1500 mAh	6:04	6:00	58%	15%	364 s	1.111
Total Error								1.222

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	*Error*
1	*Go Left*	1500 mAh	8:02	8:00	65%	15%	482 s	0.416
2	*Go Left*	1500 mAh	7:53	8:00	65%	15%	473 s	1.458
3	*Go Left*	1500 mAh	8:03	8:00	65%	15%	483 s	0.625
4	*Go Left*	1500 mAh	8:02	8:00	65%	15%	482 s	0.416
5	*Go Left*	1500 mAh	8:10	8:00	65%	15%	490 s	2.083
6	*Go Left*	1500 mAh	8:04	8:00	65%	15%	484 s	0.833
7	*Go Left*	1500 mAh	8:07	8:00	65%	15%	487 s	1.458
8	*Go Left*	1500 mAh	7:56	8:00	65%	15%	476 s	0.833
9	*Go Left*	1500 mAh	7:55	8:00	65%	15%	475 s	1.041
10	*Go Left*	1500 mAh	7:50	8:00	65%	15%	470 s	2.083
Total Error								1.125

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	*Error*
1	*Go right*	1500 mAh	6:55	7:00	60%	15%	415 s	1.190
2	*Go right*	1500 mAh	7:02	7:00	60%	15%	422 s	0.476
3	*Go right*	1500 mAh	6:59	7:00	60%	15%	419 s	0.238
4	*Go right*	1500 mAh	7:04	7:00	60%	15%	424 s	0.952
5	*Go right*	1500 mAh	7:06	7:00	60%	15%	426 s	1.428
6	*Go right*	1500 mAh	7:09	7:00	60%	15%	429 s	2.142
7	*Go right*	1500 mAh	7:13	7:00	60%	15%	433 s	3.095
8	*Go right*	1500 mAh	7:02	7:00	60%	15%	422 s	0.476
9	*Go right*	1500 mAh	6:54	7:00	60%	15%	414 s	1.428
10	*Go right*	1500 mAh	7:07	7:00	60%	15%	427 s	1.666
Total Error								1.309

Pengujian Ketepatan Estimasi Waktu Dengan input kecepatan 0.7

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	*Error*
1	*Hover*	1500 mAh	5:02	5:00	53%	15%	302 s	0.666
2	*Hover*	1500 mAh	4:59	5:00	53%	15%	299 s	0.333
3	*Hover*	1500 mAh	4:54	5:00	53%	15%	294 s	2
4	*Hover*	1500 mAh	4:55	5:00	53%	15%	295 s	1.666
5	*Hover*	1500 mAh	4:54	5:00	53%	15%	294 s	2
6	*Hover*	1500 mAh	4:56	5:00	53%	15%	296 s	1.333
7	*Hover*	1500 mAh	4:52	5:00	53%	15%	292 s	2.666
8	*Hover*	1500 mAh	5:02	5:00	53%	15%	302 s	0.666
9	*Hover*	1500 mAh	5:03	5:00	53%	15%	303 s	1
10	*Hover*	1500 mAh	5:01	5:00	53%	15%	301 s	0.333
Total Error								1.266

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	*Error*
1	*Forward*	1500 mAh	7:02	7:00	58%	15%	422	0.476
2	*Forward*	1500 mAh	6:55	7:00	58%	15%	415	1.190
3	*Forward*	1500 mAh	6:58	7:00	58%	15%	418	0.476
4	*Forward*	1500 mAh	7:12	7:00	58%	15%	432	2.857
5	*Forward*	1500 mAh	7:07	7:00	58%	15%	427	1.666
6	*Forward*	1500 mAh	7:03	7:00	58%	15%	423	0.714
7	*Forward*	1500 mAh	7:02	7:00	58%	15%	422	0.476
8	*Forward*	1500 mAh	7:08	7:00	58%	15%	428	1.904
9	*Forward*	1500 mAh	7:01	7:00	58%	15%	421	0.238
10	*Forward*	1500 mAh	7:05	7:00	58%	15%	425	1.190
Total Error								1.119

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	*Error*
1	*Backward*	1500 mAh	5:02	5:00	53%	15%	302	0.666
2	*Backward*	1500 mAh	4:51	5:00	53%	15%	291	3
3	*Backward*	1500 mAh	4:52	5:00	53%	15%	292	2.666
4	*Backward*	1500 mAh	4:59	5:00	53%	15%	299	0.333
5	*Backward*	1500 mAh	4:57	5:00	53%	15%	297	1
6	*Backward*	1500 mAh	4:55	5:00	53%	15%	295	1.666
7	*Backward*	1500 mAh	4:58	5:00	53%	15%	298	0.666
8	*Backward*	1500 mAh	4:59	5:00	53%	15%	299	0.333
9	*Backward*	1500 mAh	5:07	5:00	53%	15%	307	2.333
10	*Backward*	1500 mAh	5:09	5:00	53%	15%	309	3
Total Error								1.566

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	*Error*
1	*Go Left*	1500 mAh	6:02	6:00	55%	15%	362	0.555
2	*Go Left*	1500 mAh	6:07	6:00	55%	15%	367	1.944
3	*Go Left*	1500 mAh	6:09	6:00	55%	15%	369	2.5
4	*Go Left*	1500 mAh	5:56	6:00	55%	15%	356	1.111
5	*Go Left*	1500 mAh	5:58	6:00	55%	15%	358	0.555
6	*Go Left*	1500 mAh	5:59	6:00	55%	15%	359	0.277
7	*Go Left*	1500 mAh	6:09	6:00	55%	15%	369	2.5
8	*Go Left*	1500 mAh	6:03	6:00	55%	15%	363	0.833
9	*Go Left*	1500 mAh	6:06	6:00	55%	15%	366	1.666
10	*Go Left*	1500 mAh	6:13	6:00	55%	15%	373	3.611
Total Error								1.5555

Nomor	Gerakan	Kapasitas	Waktu	*Web*	*Start*	*End*	Waktu	*Error*
1	*Go Right*	1500 mAh	8:02	8:00	63%	15%	482	0.416
2	*Go Right*	1500 mAh	8:05	8:00	63%	15%	485	1.041
3	*Go Right*	1500 mAh	8:07	8:00	63%	15%	487	1.458
4	*Go Right*	1500 mAh	7:56	8:00	63%	15%	476	0.833
5	*Go Right*	1500 mAh	7:55	8:00	63%	15%	475	1.041
6	*Go Right*	1500 mAh	7:59	8:00	63%	15%	479	0.208
7	*Go Right*	1500 mAh	8:12	8:00	63%	15%	492	2.5
8	*Go Right*	1500 mAh	8:04	8:00	63%	15%	484	0.833
9	*Go Right*	1500 mAh	8:09	8:00	63%	15%	489	1.875
10	*Go Right*	1500 mAh	8:03	8:00	63%	15%	483	0.625
Total Error								1.083

c. Pengujian Ketepatan Estimasi Perpindahan

Percobaan Perpindahan Dengan Input kecepatan 0.3

Gerak maju

Percobaan ke-	Perpindahan (m)	waktu (s)
1	28	60
2	17	60
3	16	60
4	22	60
5	19	60
6	22	60
7	21	60
8	25	60
9	27	60
10	26	60
Rata-rata 22.3 meter		

Gerak mundur

Percobaan ke-	Perpindahan (m)	waktu (s)
1	24	60
2	36	60
3	21	60
4	40	60
5	21	60
6	44	60
7	48	60
8	21	60
9	22	60
10	27	60
Rata-rata 30.4 meter		

Gerak kiri

Percobaan ke-	Perpindahan (m)	waktu (s)
1	27	60
2	27	60
3	21	60
4	20	60
5	27	60
6	27	60

Gerak kanan

Percobaan ke-	Perpindahan (m)	waktu (s)
1	23	60
2	23	60
3	29	60
4	31	60
5	19	60
6	25	60

7	29	60
Percobaan ke-	Perpindahan (m)	waktu (s)
8	24	60
9	31	60
10	30	60
Rata –rata 26.3 meter		

7	29	60
Percobaan ke-	Perpindahan (m)	waktu (s)
8	22	60
9	29	60
10	24	60
Rata-rata 25.4 meter		

Percobaan Perpindahan Dengan Input kecepatan 0.4

Gerak maju		
Percobaan Ke-	Perpindahan (m)	Waktu (s)
1	42	60
2	57	60
3	55	60
4	56	60
5	59	60
6	41	60
7	42	60
8	52	60
9	54	60
10	54	60
Rata –rata 51.2 meter		

Gerak maju		
Percobaan Ke-	Perpindahan (m)	Waktu (s)
1	29	60
2	47	60
3	39	60
4	46	60
5	59	60
6	41	60
7	42	60
8	42	60
9	42	60
10	44	60
Rata –rata 43.1 meter		

Gerak kiri		
Percobaan Ke-	Perpindahan (m)	Waktu (s)
1	22	60
2	25	60
3	27	60
4	29	60
5	17	60
6	15	60
7	23	60
8	20	60
9	21	60
10	31	60
Rata –rata 23		

Gerak kanan		
Percobaan Ke-	Perpindahan (m)	Waktu (s)
1	20	60
2	29	60
3	23	60
4	25	60
5	17	60
6	15	60
7	20	60
8	18	60
9	19	60
10	27	60
Rata –rata 21.3		

Percobaan Perpindahan Dengan Input kecepatan 0.5

Gerak maju		
Percobaan Ke-	Perpindahan (m)	Waktu (s)
1	28	60
2	42	60
3	45	60
4	33	60
5	32	60
6	30	60
7	31	60
8	37	60
9	36	60
10	41	60
Rata –rata 35.5 meter		

Gerak maju		
Percobaan Ke-	Perpindahan (m)	Waktu (s)
1	21	60
2	32	60
3	41	60
4	33	60
5	32	60
6	30	60
7	31	60
8	37	60
9	25	60
10	41	60
Rata –rata 32.3 meter		

Gerak kiri		
Percobaan Ke-	Perpindahan (m)	Waktu (s)
1	25	60
2	29	60
3	18	60
4	21	60
5	30	60
6	24	60
7	31	60
8	22	60
9	35	60
10	27	60
Rata –rata 26.2 meter		

Gerak kanan		
Percobaan Ke-	Perpindahan (m)	Waktu (s)
1	28	60
2	32	60
3	25	60
4	28	60
5	34	60
6	27	60
7	29	60
8	25	60
9	28	60
10	25	60
Rata –rata 28.1 meter		

Percobaan Perpindahan Dengan Input kecepatan 0.6

Gerak maju		
Percobaan Ke-	Perpindahan (m)	Waktu (s)
1	42	60
2	57	60
3	55	60
4	56	60
5	59	60
6	41	60
7	42	60
8	52	60
9	54	60
10	54	60
Rata –rata 51.2 meter		

Gerak maju		
Percobaan Ke-	Perpindahan (m)	Waktu (s)
1	29	60
2	47	60
3	39	60
4	46	60
5	59	60
6	41	60
7	42	60
8	42	60
9	42	60
10	44	60
Rata –rata 43.1 meter		

Gerak Kiri		
Percobaan Ke-	Perpindahan (m)	Waktu (s)
1	53	60
2	62	60
3	65	60
4	56	60
5	61	60
6	44	60
7	45	60
8	49	60
9	63	60
10	65	60
Rata –rata 56.3 meter		

Gerak Kanan		
Percobaan Ke-	Perpindahan (m)	Waktu (s)
1	31	60
2	40	60
3	41	60
4	39	60
5	51	60
6	44	60
7	38	60
8	37	60
9	40	60
10	41	60
Rata –rata 40.2 meter		

Percobaan Perpindahan Dengan Input kecepatan 0.7

Gerak maju		
Percobaan Ke-	Perpindahan (m)	Waktu (s)
1	72	60
2	65	60
3	41	60
4	43	60
5	57	60
6	41	60
7	40	60
8	47	60
9	46	60
10	40	60
Rata –rata 49.2 meter		

Gerak maju		
Percobaan Ke-	Perpindahan (m)	Waktu (s)
1	40	60
2	63	60
3	50	60
4	52	60
5	51	60
6	49	60
7	45	60
8	52	60
9	47	60
10	65	60
Rata –rata 51.4 meter		

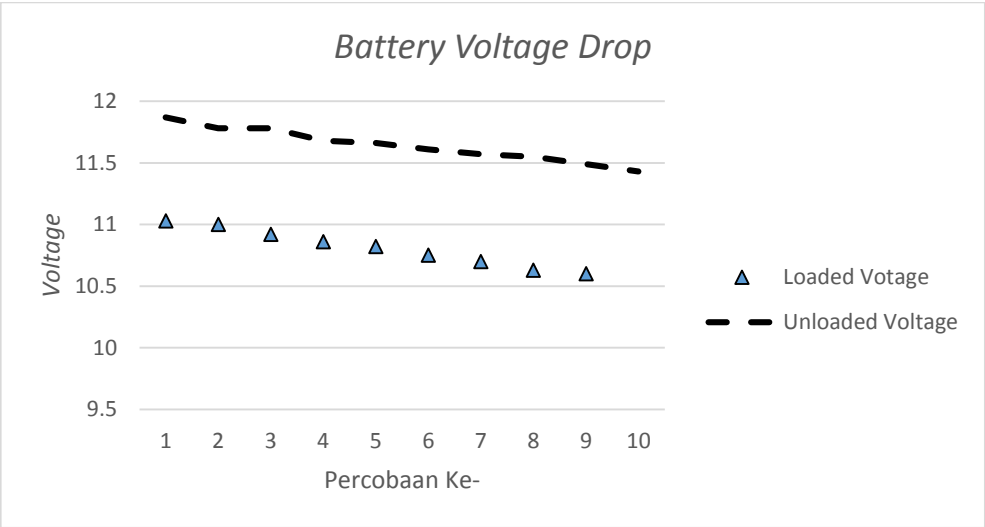
Gerak kiri		
Percobaan Ke-	Perpindahan (m)	Waktu (s)
1	44	60
2	57	60
3	59	60
4	58	60
5	59	60
6	44	60
7	46	60
8	49	60
9	50	60
10	47	60
Rata –rata 51.3 meter		

Gerak kanan		
Percobaan Ke-	Perpindahan (m)	Waktu (s)
1	87	60
2	51	60
3	47	60
4	43	60
5	53	60
6	41	60
7	40	60
8	42	60
9	43	60
10	46	60
Rata –rata 49.3 meter		

d. Percobaan *Voltage Drop*

Tabel Percobaan *Internal Resistance* pada Baterai *AR.Drone 2.0*

Capacity (mAh)	Cell	Run Time (s)	Unloaded Voltage (v)	Current (mA)	Loaded Voltage (v)	Voltage Drop (v)	Resistance (ohm)	Power (Wh)
1500	3	17	11.87	1.07	11.08	0.87	813	16.6
1500	3	17	11.78	1.07	11.03	0.75	701	16.6
1500	3	17	11.78	1.07	11	0.86	804	16.6
1500	3	17	11.68	1.06	10.92	0.82	774	16.6
1500	3	17	11.66	1.1	10.86	0.58	527	16.6
1500	3	17	11.61	1.08	10.82	0.79	731	16.6
1500	3	17	11.57	1.07	10.75	0.82	766	16.6
1500	3	17	11.55	1.06	10.7	0.85	802	16.6
1500	3	17	11.49	1.05	10.63	0.86	819	16.6
1500	3	17	11.43	1.05	10.6	0.83	790	16.6



Gambar *Voltage Drop* Pada Baterai *AR.Drone 2.0*

e. Informasi komponen AR.Drone 2.0

Tabel Komponen *AR.DRONE 2.0*

Number	*Components*	*Voltage Recommendation*	*Current (mA)*	*Resistance Load* (Ω)	*Thevenin Resistance* (Ω)	*Power (Watt)*
1	*Atheros AR16310*					
	AVDD18	1.8	R(8.2) | T(58)	R(0.3 K) | T(53.44)	R(0.882 K) | T(0.135 K)	R(0.014) | T(0.104)
	PAREG_BASE	3.3	R(0) | T (175)	R(0) | T (17.71)	R(0) | T(0.135 K)	R(0) | T(0.57)
	VAT_42	3	R(0) | T (175)	R(0) | T (17.71)	R(0) | T(0.135 K)	R(0) | T(0.57)
	VDD33	3	R(0) | T (175)	R(0) | T (17.71)	R(0) | T(0.135 K)	R(0) | T(0.57)
2	*TPS65921*	4	1200	2.25	7.0	4.32
3	*NAND Flash 128*	3	8	337.5	16.81	2.4×10^{-5}
4	*MT46H64*	2	46	36.95	203.25	0.082
5	*BMP180*					
	VDD	3	0.005	360 K	1.83 M	1.35×10^{-8}
	VDD IO	2.6	0.005	324 K	1.87 M	1.35×10^{-8}
6	*IMUU-3000*	3	65	32.30 K	0.135 K	0.18
7	*PIC24HJ*	3.3	250	12	32.41	0.82
8	*BM150*					
	VDD	3	200	12	43.24	0.6
	VDD IO	2	200	8.1	46.98	0.44
9	*MCP6L04*	3.3	2	900	4.4 K	0.0066
10	*AK8975C*					
	Analog Power Supply	3	0.35	6.8 K	24.685 K	1.05×10^{-6}
	Digital InterFace	2	0.35	4.7 K	26.865 K	5.7×10^{-7}

Sumber: Teardown.com (2016)